# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATE COVERED |
|---|---|---|
| | 3 May 2007 | |

**4. TITLE AND SUBTITLE**
Autonomous Detection and Imaging of Abandoned Luggage in Real World Environments

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**
**Papon, Jeremie A.**

| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) | | 8. PERFORMING ORGANIZATION REPORT NUMBER |
|---|---|---|
| | | |

| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
|---|---|---|
| **US Naval Academy<br>Annapolis, MD 21402** | | **Trident Scholar project report no. 357 (2007)** |

**11. SUPPLEMENTARY NOTES**

| 12a. DISTRIBUTION/AVAILABILITY STATEMENT | 12b. DISTRIBUTION CODE |
|---|---|
| **This document has been approved for public release; its distribution is UNLIMITED.** | |

**13. ABSTRACT** . This Trident Project developed a system that is able to detect and produce high resolution imagery of unattended items in a crowded scene, such as an airport, using live video processing techniques. Video surveillance is commonplace in today's public areas, but as the number of cameras increases, so do the human resources required to monitor them. Additionally, current surveillance networks are restricted by the low resolution of their cameras. For example, while there is an extensive security camera network in the London Underground, its low resolution prevented it from being used to automatically identify the terrorists that entered the train stations in July 2005. With this in mind, this project developed a surveillance system that is able to autonomously monitor a scene for suspicious events by combining a low resolution camera for surveillance (a webcam) with a moving high resolution camera (a 6 mega-pixel digital still-frame camera) to provide a greater level of detail. This enhanced capability is used to determine whether or not the event is a threat. For the purposes of this research, suspicious events were defined as a person leaving a piece of luggage unattended for an extended period of time. Initial analysis of the surveillance video involved separating the foreground (such as people carrying luggage) from the background. In order to do this using live video, an automated algorithm was developed which creates a composite background image from a small number of video frames.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Biometrics, Motion Detection, Security Camera, Surveillance, Video | **82** |
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| | | | |

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATE COVERED |
|---|---|---|
| | 3 May 2007 | |

**4. TITLE AND SUBTITLE**
Autonomous Detection and Imaging of Abandoned Luggage in Real World Environments

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**
**Papon, Jeremie A.**

| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) | | 8. PERFORMING ORGANIZATION REPORT NUMBER |
|---|---|---|
| | | |

| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
|---|---|---|
| **US Naval Academy<br>Annapolis, MD 21402** | | **Trident Scholar project report no. 357 (2007)** |

**11. SUPPLEMENTARY NOTES**

| 12a. DISTRIBUTION/AVAILABILITY STATEMENT | 12b. DISTRIBUTION CODE |
|---|---|
| **This document has been approved for public release; its distribution is UNLIMITED.** | |

**13. ABSTRACT** (CONT.)In the algorithm, areas detected as motion were removed from individual frames. These processed frames, which represented regions of no motion, were then combined by taking the median value for each pixel across all frames. These median values were used to form a composite background image which contained only the non-moving parts of the scene (i.e., the background). Once this background was obtained, the system then detected live motion. Using a variety of filtering techniques, individual foreground objects were separated from the stationary background. These objects were then tracked over time. When an object (for the purposes of this research, a moving object was assumed to be a person) divided into two different objects, they were then tracked to see if one of the objects remained motionless. In doing so, the system was able to detect an "abandonment" event. When such an event occurred, an event timer began to determine how long the luggage had been abandoned. If the luggage was left unattended for a preset amount of time, the system tagged it for high resolution imaging. Once an abandoned item was tagged for high resolution imaging, the system used a motorized pan/tilt mount to point the high resolution camera and acquire a high resolution image of the item. This image was then sent to a human supervisor for further investigation. The final security system can allow a single person to monitor a vast array of camera systems (spanning for example, an entire airport) for abandoned luggage or any other pre-defined suspicious event

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Biometrics, Motion Detection, Security Camera, Surveillance, Video | **82** |
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| | | | |

## Project Abstract

This Trident Project developed a system that is able to detect and produce high resolution imagery of unattended items in a crowded scene, such as an airport, using live video processing techniques. Video surveillance is commonplace in today's public areas, but as the number of cameras increases, so do the human resources required to monitor them. Additionally, current surveillance networks are restricted by the low resolution of their cameras. For example, while there is an extensive security camera network in the London Underground, its low resolution prevented it from being used to automatically identify the terrorists that entered the train stations in July 2005. With this in mind, this project developed a surveillance system that is able to autonomously monitor a scene for suspicious events by combining a low resolution camera for surveillance (a webcam) with a moving high resolution camera (a 6 mega-pixel digital still-frame camera) to provide a greater level of detail. This enhanced capability is used to determine whether or not the event is a threat.

For the purposes of this research, suspicious events were defined as a person leaving a piece of luggage unattended for an extended period of time. Initial analysis of the surveillance video involved separating the foreground (such as people carrying luggage) from the background. In order to do this using live video, an automated algorithm was developed which creates a composite background image from a small number of video frames. In the algorithm, areas detected as motion were removed from individual frames. These processed frames, which represented regions of no motion, were then combined by taking the median value for each pixel across all frames. These median values were used to form a composite background image which contained only the non-moving parts of the scene (i.e., the background).

Once this background was obtained, the system then detected live motion. Using a variety of filtering techniques, individual foreground objects were separated from the stationary background. These objects were then tracked over time. When an object (for the purposes of this research, a moving object was assumed to be a person) divided into two different objects, they were then tracked to see if one of the objects remained motionless. In doing so, the system was able to detect an "abandonment" event. When such an event occurred, an event timer began to determine how long the luggage had been abandoned. If the luggage was left unattended for a preset amount of time, the system tagged it for high resolution imaging.

Once an abandoned item was tagged for high resolution imaging, the system used a motorized pan/tilt mount to point the high resolution camera and acquire a high resolution image of the item. This image was then sent to a human supervisor for further investigation. The final security system can allow a single person to monitor a vast array of camera systems (spanning for example, an entire airport) for abandoned luggage or any other pre-defined suspicious event.

**Keywords: Biometrics, Motion Detection, Security Camera, Surveillance, Video**

# Acknowledgments

The author wishes to express his appreciation to:

Dr. Robert Ives, Electrical Engineering Department, USNA – Primary Project Adviser

Dr. Randy Broussard, Weapons and Systems Engineering Department, USNA –Secondary Project Adviser

Mr. Jeffery Dunn, National Security Agency –External Collaborator

Ms. Daphi Jobe, Electrical Engineering Department, USNA- Electronics Technician

Mr. Jerry Ballman, Electrical Engineering Department, USNA- Laboratory Technician

Mr. Michael Wilson, Electrical Engineering Department, USNA – Laboratory Technician

# Table of Contents

# Table of Figures

# 1. The Surveillance Problem

Video surveillance is extremely common in modern public areas, and as the number of cameras increases, so do the number of video feeds to monitor. This places an increasing burden on security personnel, as more and more must be assigned to the task of monitoring the videos. Additionally, a person is only able to watch so many video feeds at the same time with any sort of efficiency, and as with any system involving human monitors, are subject to human error. Take for example the London Subway Bombings of July 7th 2005, in which the terrorists passed directly in front of several security cameras (Figure 1). Even though the terrorists were on several watch lists, and security personnel were in a heightened state of alert, the terrorists were allowed free access to the subways, and managed to leave their bags (which contained bombs) throughout the London Underground.


**Figure 1: Terrorists in London Subway on July 7th 2005 (United Kingdom Crown Copyright, available at http://news.bbc.co.uk/2/hi/uk_news/politics/4689739.stm)**

In addition to the fallibility of human monitors, current surveillance networks suffer from another major drawback; their low resolution. This severely limits the ability to identify threats via automated algorithms, such as the one presented in this report. The low resolution of current security systems may also prevent the implementation of covert automated facial recognition of people which might have been used to alert security personnel to the presence of terrorists in the London Underground. As such, this research has developed a dual camera system, which combines a low resolution video camera with a high resolution still-frame camera mounted on a moving pan/tilt mount. Combined with supporting application software, this allows for automated analysis of video to detect objects of interest and investigate them using the high resolution camera.

## 2. System Components

The low resolution camera is a Logitech Quickcam Pro 5000 (Figure 2). This camera has proven ideal for the purpose of surveillance because it combines good resolution (640 x 480 pixels), fast frame rates (up to 30 frames per second), and a wide field of view (82 degrees) in a small robust package. In this system, the Quickcam is stationary, and surveys an entire area of interest at once, continuously monitoring for movement. When movement in the field of view is detected, the camera video output is processed for object localization. The Quickcam interfaces directly with MATLAB (a high-performance programming language for technical computing) running on a Personal Computer (PC) via a Universal Serial Bus (USB) cable, which allows for quick processing of the video feed.

**Figure 2: Logitech Quickcam Pro 5000 (from Logitech at http://www.logitech.com/)**

The high resolution camera consists of a 6 mega-pixel (a mega-pixel is one million pixels) Canon Powershot S3IS (Figure 3). This camera was chosen because of its good low-light performance and motorized zoom. The twelve-times optical zoom is extremely fast, and can be controlled remotely, allowing for imaging of small objects at relatively long distances. The camera is mounted on a high speed pan/tilt mount for pointing, and comes with a Software Development Kit (SDK) which allows for remote control of the camera over a USB cable.


**Figure 3: Canon PowerShot S3IS (from Canon USA at http://www.usa.canon.com/)**

A pan/tilt mount is a device that can interpret input digital signals to control its position in all three dimensions. This project uses a Directed Perception PTU-D46-17 pan/tilt mount which can pan (left/right) a full 360 degrees and tilt (up/down) 180 degrees, at speeds of over 300 degrees per second. The mount (Figure 4) allows for fast, stable, and accurate pointing of the high resolution camera. The pan/tilt mount is controlled through the serial port of the PC, and can be commanded directly from a Graphical User Interface (GUI) using the serial port

functionality built into MATLAB. The GUI portion of the system and its control of the system's hardware components is described in the next section.

**Figure 4: Directed Perception Pan/Tilt Mount (from Directed Perception at http://www.dperception.com/)**

# 3. Interfacing of Components

The first major hurdle in developing a system that combines various off-the-shelf components into a single working unit is interfacing. Both the high resolution camera and the pan/tilt mount required separate controls to be developed in order to meet their desired functionality. This was accomplished in MATLAB.

Control of the Pan/Tilt mount was accomplished using the computer's serial port. The serial port is used to send commands from MATLAB to the pan/tilt mount's controller, and allows control of both the movement of the mount and all of its settings, such as slew rate (maximum turning rate) and acceleration. The commands to move the pan/tilt mount which serve to point the high resolution camera are based on the location of the objects in the view of the low resolution webcam.

Controlling the Canon Powershot high resolution camera proved to be much more difficult. The primary reason for this is that the Canon SDK, which provides a basic framework for communicating with their cameras, was written in the C programming language. In order to

use the camera, all of the control functions had to be ported into MATLAB using MEX files. MEX files are a specific type of MATLAB file which are written in C, but can compile and run in MATLAB. MEX files for initializing the camera, changing camera settings, and controlling the shutter were created. But the actual transfer of images from the high resolution camera to the computer over the USB cable proved problematic, as the Canon framework for doing this could not be ported into MATLAB. As such, new functions were written in C which transferred the image data from the camera over the USB cable in individual packets instead of one large file transfer. Packet sizes in initial testing were limited to only 1024 bytes, which resulted in very poor transfer speeds for images (upwards of 5 seconds). After several adjustments and revisions to the transfer protocol, 128 kilobyte packets were used successfully, which increased transfer speeds substantially (approximately one second for a one megabyte image). The complete assembled system can be seen in Figure 5.


**Figure 5: The Assembled System**

The entire system is controlled using a MATLAB GUI that was compiled into a standard Windows executable file for faster execution speed. The GUI, shown in Figure 6, allows a human operator to control all of the system components using a single interface. It allows the operator to adjust settings for each component, view the low resolution video feed, view a graphical representation of the object tracking algorithm, and view any high resolution images acquired. The GUI has built-in options to allow for the control of several camera systems at once; this allows for the control of an entire surveillance network from one central computer. A detailed description of the GUI's functionality is contained in Appendix A, and the algorithms that run the controls are described in the following sections.

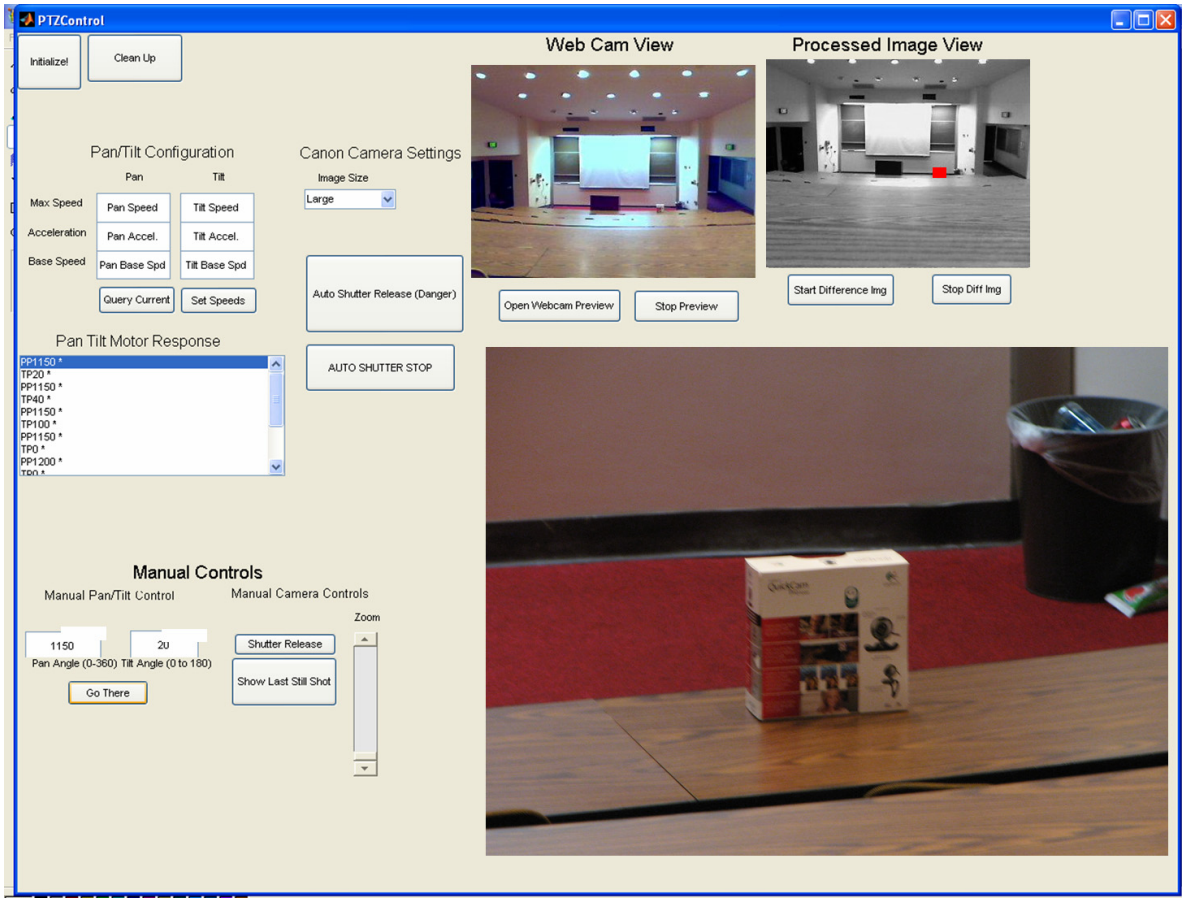


**Figure 6: Graphical User Interface**

# 4. Composite Background Creation

One approach to detecting abandoned luggage in a video frame is to compare the frame to an image of the same scene that has no abandoned luggage in it (the background). The first element necessary for accurate analysis of a live surveillance feed was therefore the creation of an accurate background image. This is easy in a laboratory, where one can simply clear the area of people and other foreground objects. Unfortunately, in real world applications it can be difficult, if not impossible, to clear an area in order to get a background image. Additionally, even if one clears an area and obtains an accurate background image when a camera is installed, any changes in the background (such as a light burning out, or furniture being moved) can severely degrade the system's ability to separate foreground from background. There are several methods that have been developed to create composite backgrounds, most common of which are Kalman filtering [1] and mixture of Gaussians [2]. For the purposes of this research, none of these techniques were computationally efficient enough for real-time applications, so a new technique was developed that is far more efficient while still attaining equal, if not better, levels of accuracy. With this in mind, this research has developed a robust algorithm which is able to take live video and dynamically create a "composite" background image (Figure 7). Using this algorithm, a background can be created even with people present in the field of view of the camera. Additionally, because the algorithm works quickly and can be applied to live video, it can be used to periodically update the background image automatically to account for any changes to the background.

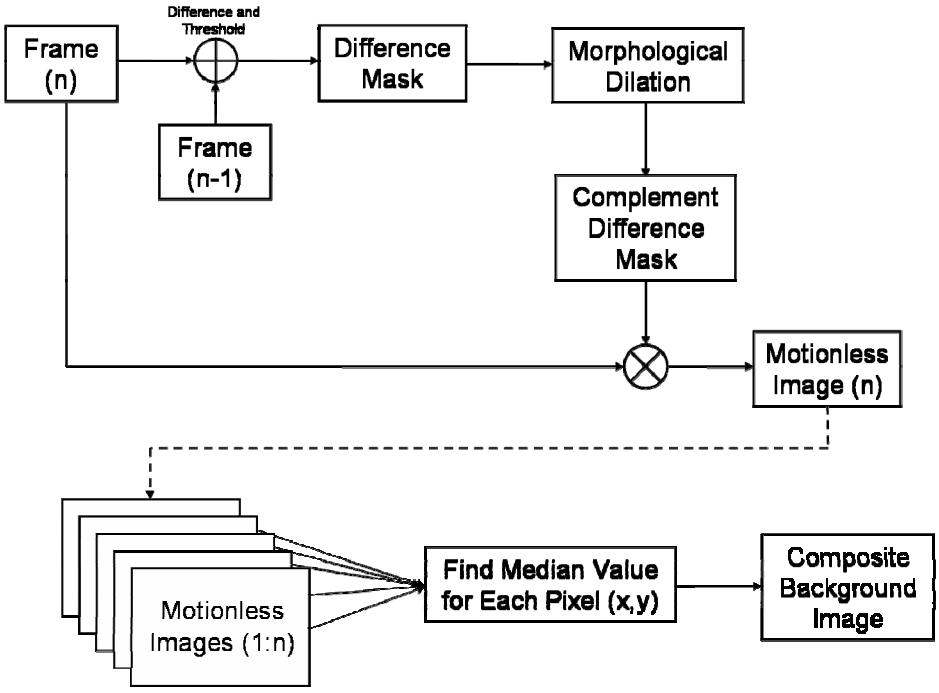


**Figure 7: Composite Background Creation Algorithm**

The first step in the background creation is simple frame-to-frame differencing and thresholding to detect regions that contain movement. Thresholding creates a binary image (or mask) with values of zero or one in every pixel location, where zero corresponds to motion and one to motionless (background). Threshold values for creation of the binary image are calculated using Otsu's method [3]. This difference mask is then processed with binary morphology (dilation) to account for noise in the thresholding and reduce the possibility of missing actual motion. Noise occurs naturally, since lighting conditions vary continually over time, even when humans perceive constant illumination. Under electrical lighting, this is due in variations in the electric power source. The mask is then multiplied by the current video frame to create a motionless image frame, which contains only background areas, and is then stored. This process is repeated until every pixel location has at least five "motionless" values. Once the array of

motionless images is created the median value for each pixel is calculated, and used to form the final composite background image. In testing this composite background creation process was shown to take generally about 4 seconds (at 15 frames per second) for normal surveillance video clips. An example of the background creation is shown in Figure 8. The surveillance clip used comes from a London train station, filmed for the "Performance Evaluation of Tracking and Surveillance Conference 2006" (PETS 2006) [4]. The composite background image is then used in detecting and segmenting the foreground for follow-on video.

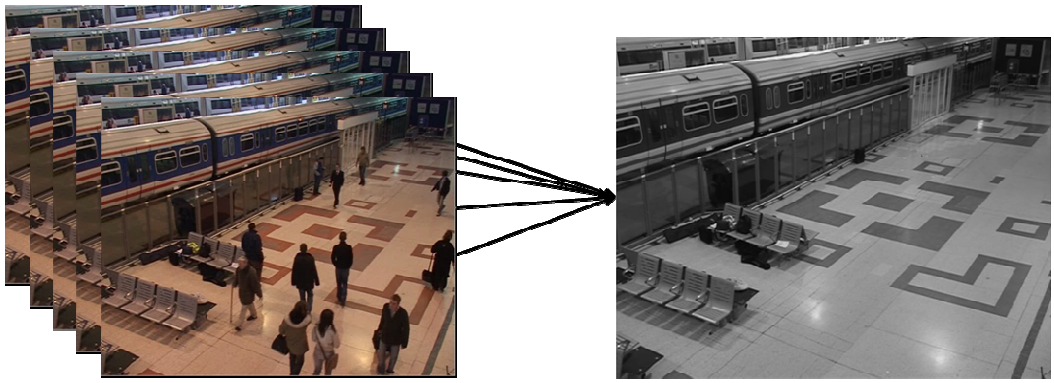

**Figure 8: Example of Background Creation Using Live Video Frames**

# 5. Foreground Segmentation

Once a composite background image is created, foreground segmentation is possible. This is accomplished using the algorithm shown in the block diagram of Figure 9. The first step in the process is differencing and thresholding both the current video frame and the previous frame with the background image to create binary masks, where binary 1 represents regions not in the background. These masks are then filtered using a majority filter and then morphologically closed. This majority filter operates in a 3-by-3 neighborhood around each pixel in each binary mask, and sets that pixel to a value of one if there are five or more ones in its neighborhood (otherwise it sets the pixel to zero) [5]. These two different masks (derived from the current and

previous video frames) are then combined using a logical AND function to create a foreground mask.

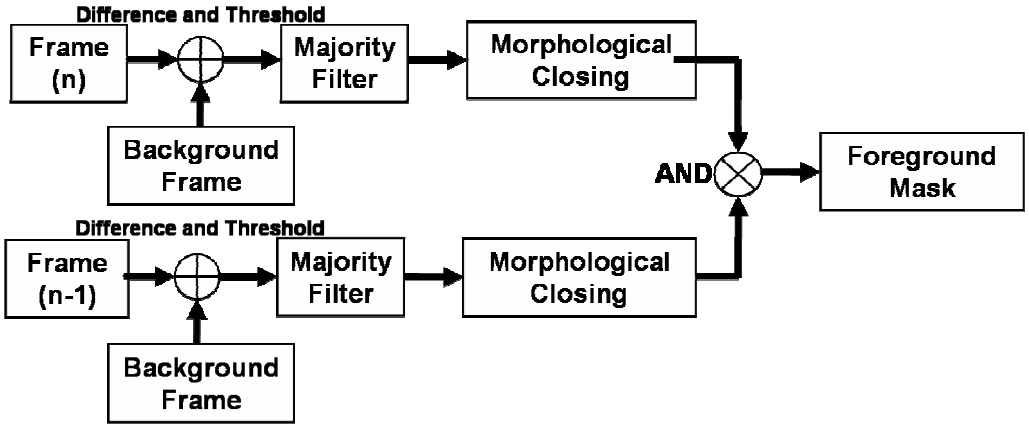


**Figure 9: Foreground Segmentation Algorithm**

This process of combining the current frame with the previous frame was shown to be extremely effective in testing, and resulted in more accurate segmentation results than could be attained by using only the current frame. An example of the foreground segmentation process applied to the PETS 2006 data set is shown in Figure 10. Once the foreground mask is obtained, the algorithm moves on to the next stage, object tracking.

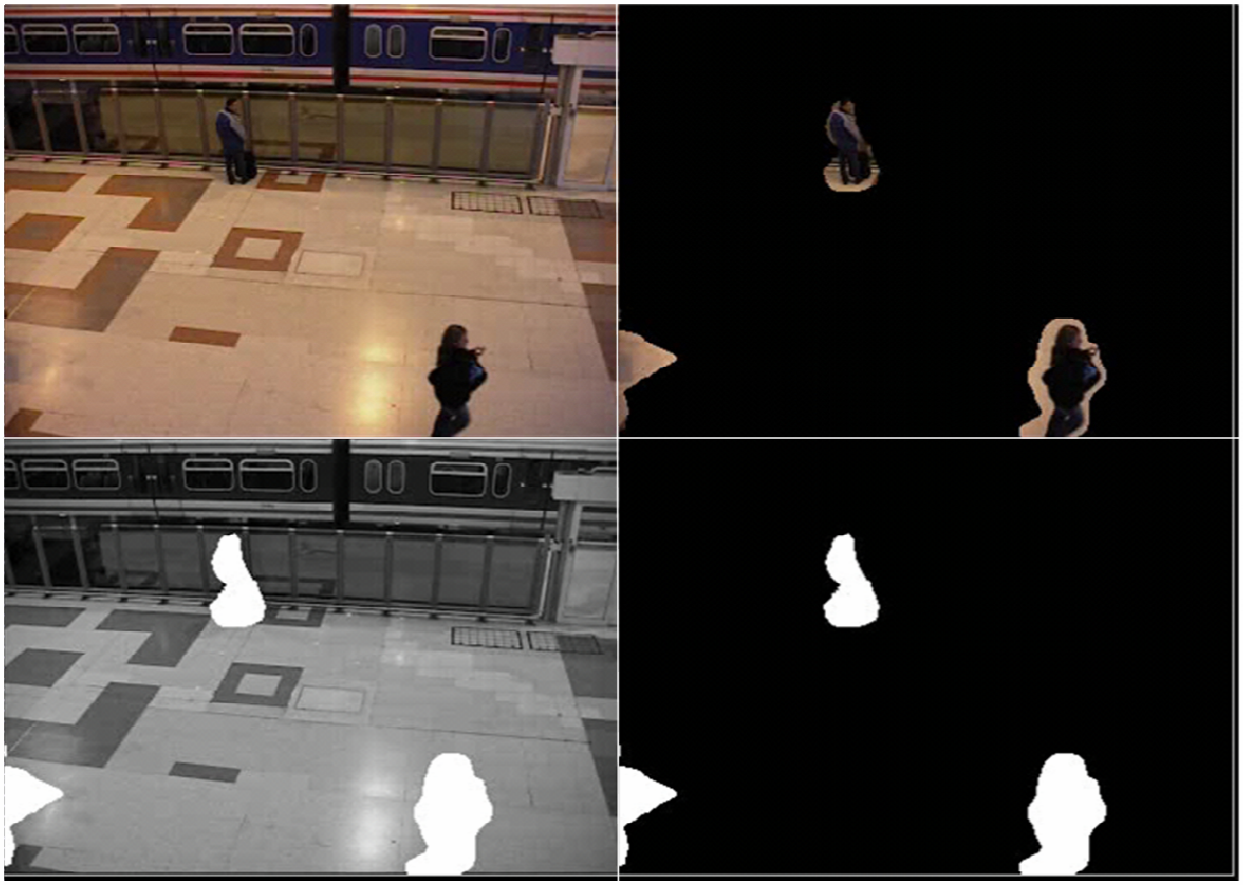

**Figure 10: Example of Foreground Segmentation in London Train Station**

**Upper Left:** Input video frame.
**Upper Right:** Cutout showing foreground segmented.
**Lower Left:** Foreground mask applied to background frame.
**Lower Right:** Binary foreground mask.

# 6. Object Tracking

The final stage of the algorithm (shown in Figure 11) is the actual tracking of foreground objects. Foreground objects consist of regions of contiguous binary 1's in the foreground mask. The size and center of each of these regions is calculated, and compared with the objects present in the previous iteration. If an object is "new", it is stored in an object structure. An object structure is a block of data which contains all the pertinent information about a particular object, such as size, location and speed. In testing, new objects were detected when either people entered

the scene, a group of people separated, or a person and their luggage separated. If an object was present in the previous frame and is present in the current frame, its statistics are updated and its motion vector (direction and speed) is calculated. This motion vector is used to determine if an object is idle; if the object remains idle for longer than a threshold time *t*, it is queued for imaging using the high resolution camera. For this research, the threshold time *t* was defined as 15 seconds.

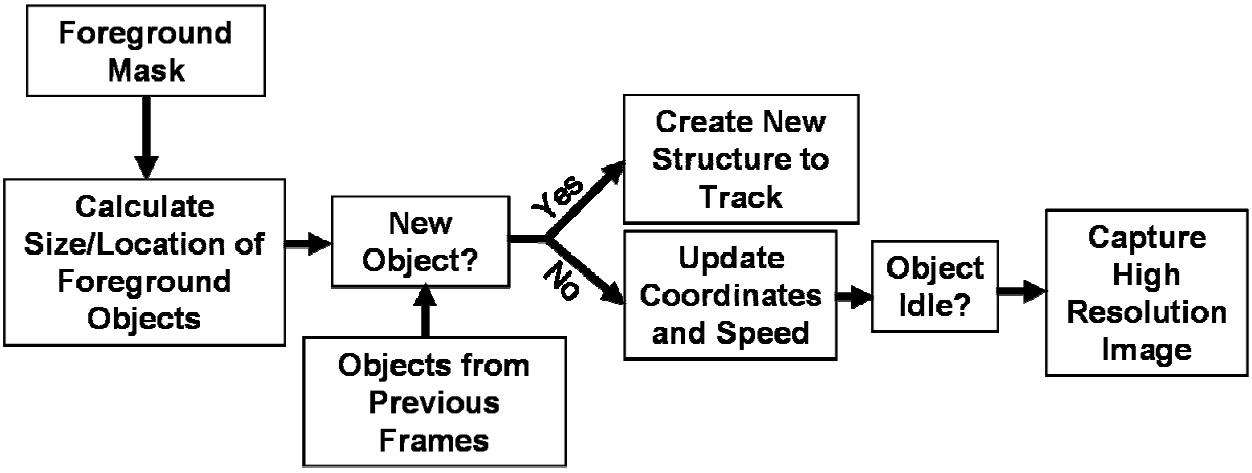


**Figure 11: Foreground Object Tracking**

In testing, it was shown that people standing still were rarely detected as "idle" because even if a person is not moving, they are seldom, if ever, perfectly still. Abandoned luggage on the other hand was detected with a great degree of reliability, because unlike people, it is perfectly motionless for significant amounts of time. When an object is queued for high resolution imaging, its size is used to determine the zoom applied to the high resolution camera, and its coordinates are passed to the pan/tilt mount to point the camera. An example of a frame from the object tracking process is shown in Figure 12.

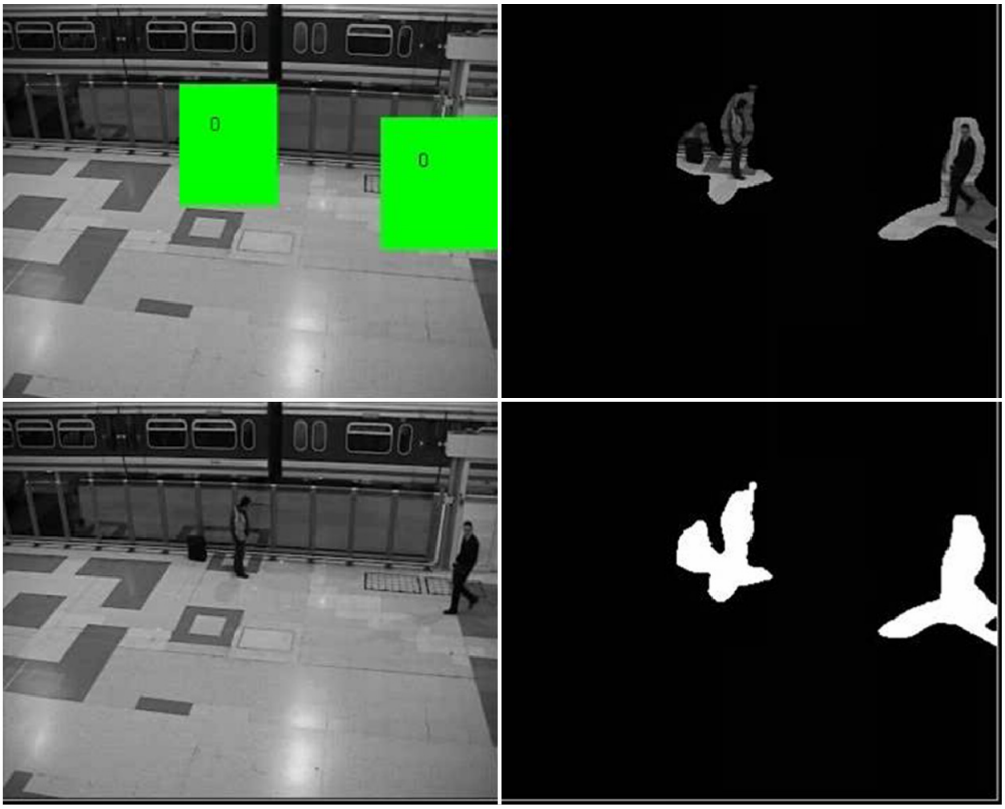

**Figure 12: Example of Object Tracking**

**Upper Left:** Visual representation of object tracking, non-idle objects shown as green boxes.
**Upper Right:** Cutout showing foreground segmented.
**Lower Left:** Input video frame.
**Lower Right:** Foreground mask

# 7. Testing

The first testing conducted was of the background creation algorithm, using previously recorded video. Five test scenario movie clips were used, three from the PETS2006 database [2] and two from video of public areas recorded at the United States Naval Academy. Each of these test scenarios was processed 10 times, with the first test conducted starting at the beginning of the clip, the second at 10 seconds, the third at 20 seconds, and so on. The first test was used to determine the time it took to calculate the composite background image from the motionless

images using either an averaging or median filter. As can be seen in Figure 13, the median algorithm took longer in all five scenarios.

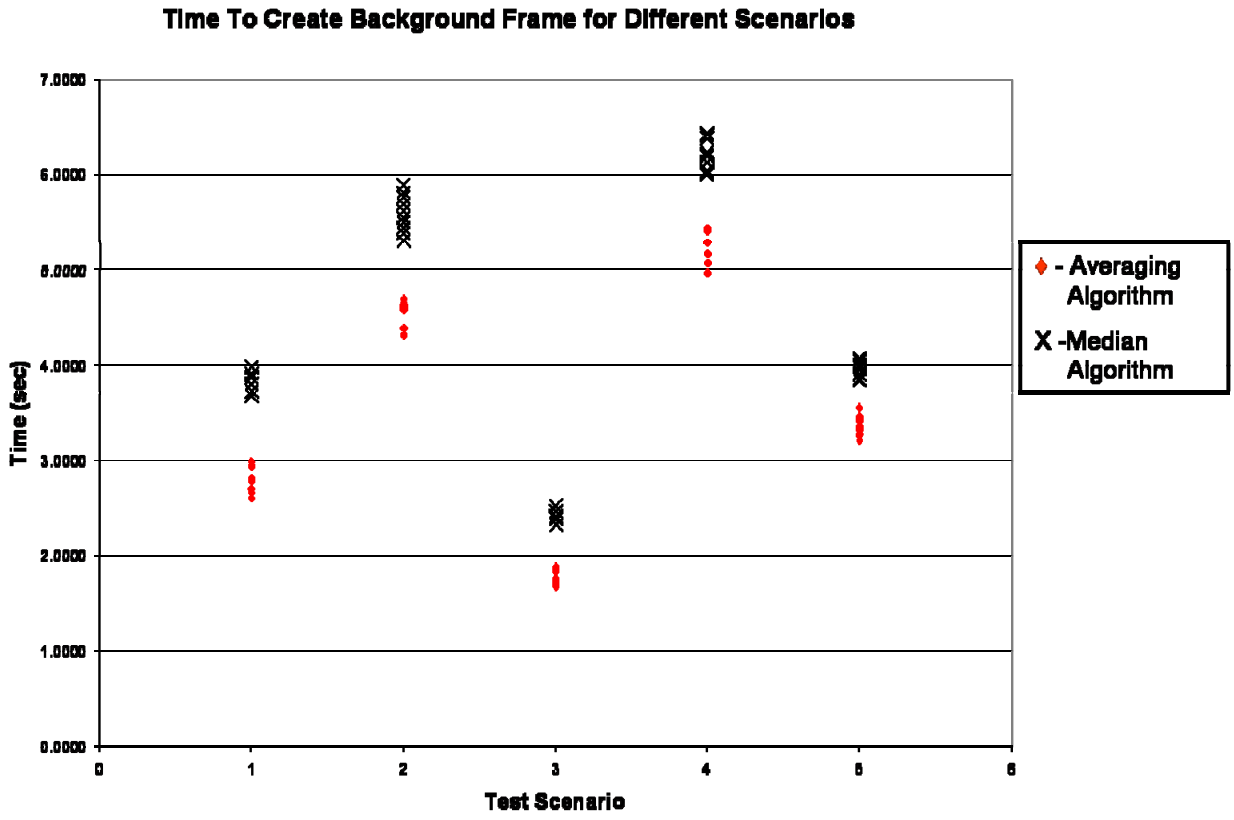


**Figure 13: Time to Create Composite Background Images**

The next tests focused on the accuracy of the background images created in the previous testing. Because this research used pre-recorded clips, it was possible to obtain ideal background images; each clip contained at least one frame where there were no foreground objects present in the field of view. By using these frames as reference, it was possible to test the accuracy of each background image created by taking the absolute error of each pixel in the composite backgrounds, and averaging them. This accuracy is displayed in Figure 14, which shows the average error for the median and averaging algorithms in the five scenarios. It is readily apparent that the median algorithm is significantly more accurate in all cases.

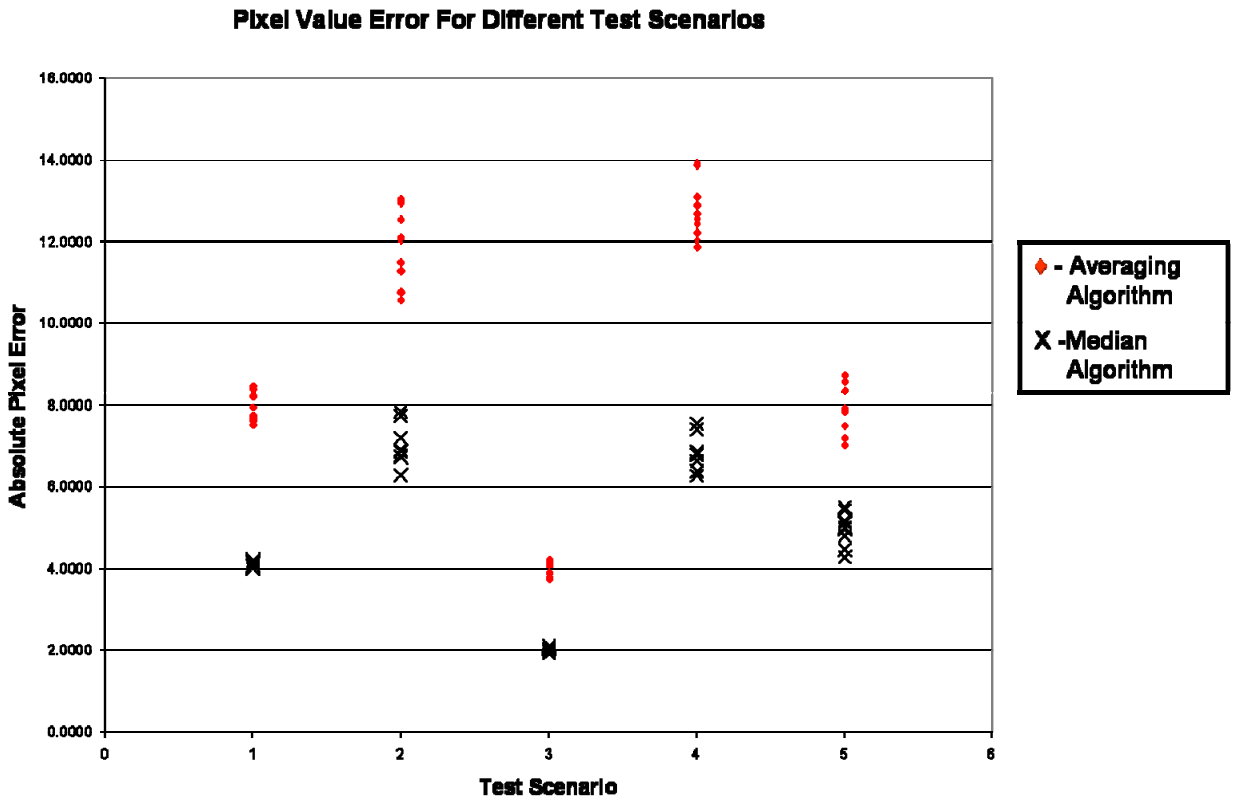


**Figure 14: Accuracy of Background Creation Algorithms**

The final stage of testing involved the complete system in staged scenarios. Two testing sessions were conducted at the United States Naval Academy, the first in a lecture hall (Rickover 103) and the second in an auditorium (Mahan Hall Auditorium). The purpose of the test in the lecture hall was to determine performance in poor lighting conditions. The auditorium test was done to show performance in good lighting conditions, which were achieved by using stage lights.

Both testing sessions followed the same general procedure. The system was set up and subjects were instructed to enact several different scenarios. These scenarios ranged from a crowd of people walking back and forth, with no baggage abandoned, to having everyone in the crowd abandon some type of baggage. Testing results are shown in Figure 15. The quantities

measured were: “Correct Detection Rate”; “Average Frames to Acquire Object(s) as Idle”; and “False Detections” per test run. The “Correct Detection Rate” is a measure of the percentage of abandoned items the system successfully captured in a high resolution image. “Average Frames to Acquire Object(s) as Idle” is a measure of the amount of time it took for the system to identify that an object has been abandoned (the system ran at 15 frames per second). “False Detections” are defined as the system queuing a foreground object for high resolution imaging when it was in fact not abandoned luggage.

| LOW LIGHT CONDITIONS | Number of Tests Run | Correct Detection Rate | Average Frames to Acquire Object(s) as Idle | False Detections per Test Run |
|---|---|---|---|---|
| **Nothing Idle, People Wandering** | 2 | N/A | N/A | 1 |
| **1 Person, 1 Drop** | 2 | **100.00%** | 12 | 0 |
| **Multiple People, 1 Drop** | 1 | **100.00%** | 10 | 0 |
| **2 People 2 Drops** | 1 | **100.00%** | 14.5 | 0 |
| **Multiple People, 2 Drops** | 1 | **100.00%** | 15.75 | 1 |
| **3 People, 3 Drops** | 1 | **100.00%** | 17 | 0 |
| **Multiple People, 3 Drops** | 1 | **66.66%** | 17.25 | 1 |
| **Multiple People, 8 Drops** | 1 | **62.50%** | 19 | 1 |
| **GOOD LIGHT CONDITIONS** | **Number of Tests Run** | **Correct Detection Rate** | **Average Frames to Acquire Object(s) as Idle** | **False Detections per Test Run** |
| **Nothing Idle, People Wandering** | 2 | N/A | N/A | 0.5 |
| **1 Person, 1 Drop** | 3 | **100.00%** | 8 | 0 |
| **Multiple People, 1 Drop** | 3 | **100.00%** | 9 | 0 |
| **2 People 2 Drops** | 2 | **100.00%** | 12 | 0 |
| **Multiple People, 2 Drops** | 2 | **100.00%** | 14 | 0.5 |
| **3 People, 3 Drops** | 3 | **100.00%** | 15.33 | 0 |
| **Multiple People, 3 Drops** | 2 | **100.00%** | 14 | 0.5 |
| **Multiple People, 8 Drops** | 2 | **87.50%** | 16 | 0.5 |

**Figure 15: Integrated System Testing Results**

At the same time, the high resolution images obtained were visually compared to determine the effect of lighting conditions on image quality. A sample of this comparison is shown in Figure 16 and Figure 17. For reference, these photographs were taken from a distance of approximately 50 feet. As can be seen, the low lighting hampered the camera's ability to focus accurately. Some of the blurring apparent in Figure 16 is also due to the slower shutter speed needed because of the poor lighting.

**Figure 16: High Resolution, Poor Lighting Conditions**


**Figure 17: High Resolution, Good Lighting Conditions**

# 8. Conclusions

One of the primary difficulties in working with an automated security system in the real world is that variations in the scenery can cause significant errors. Creating an algorithm that is able to adapt itself to any environment presented to it is one of the biggest hurdles to overcome in order to create a reliable autonomous security system. This research successfully accomplishes this with the background creation algorithm. The system's ability to rapidly create an accurate representation of an "empty" scene allows deployment at any location and immediate use. Additionally, the use of temporal differencing in the foreground segmentation algorithm proved extremely successful for creating an accurate foreground mask.

The high resolution camera system proved to be dependable and effective in testing, allowing for distant objects that were barely visible in the low resolution video to be photographed in great detail. While the objects could not usually be identified in the low resolution (webcam) video feed, the high resolution photos could allow a human supervisor to determine whether or not they consist of abandoned luggage even at long range. A surveillance system could easily be set up using any number of the high/low resolution camera systems, covering a large area (such as an entire airport or train station) and monitored by only a single supervisor, since they would only need to occasionally examine the high resolution photographs flagged as security risks, rather than continuously viewing surveillance videos.

While the images taken under poor lighting conditions were somewhat out of focus and dark, they were still clear enough for an operator to positively identify the object being photographed. The images taken in good lighting conditions on the other hand have excellent resolution, as can be seen in Figure 17. This photograph, taken from a distance of approximately fifty feet, actually allows a human operator to read text on the object. For comparison purposes,

the same object as seen in the low resolution video feed from the webcam was nothing more than a fuzzy spot, barely ten pixels across. The problems seen in the high resolution images that were caused by poor lighting could be solved by using a true single-lens reflex (SLR) camera, which, due to the quality of its optics, would have better low-light imaging capability.

System errors in testing consisted of false detections and missed detections. There were two sources of these errors that were encountered. The first was system saturation; this occurred in the scenarios involving eight people abandoning luggage. The system was simply unable to keep up with the demand put on it by having to track so many people and abandoned items. As can be seen in the results table of Figure 15, system saturation occurred with fewer test subjects and had a more detrimental effect under low light conditions. In terms of practical applications of the system, this saturation is not a significant issue, as it is highly unlikely that one would ever encounter a scenario where eight or more bags were being abandoned simultaneously.

The second type of error, false detection, is more significant. False detection errors were primarily caused by large fluctuations in background lighting. This was primarily seen in the testing done under poor lighting conditions, due to the fact that when there is not much light to begin with, any variation will be perceived as significant. While the algorithms used were designed with this in mind, and were for the most part successful, some errors did occur. These errors would occasionally result in the foreground segmentation algorithm detecting objects that were not actually present. This resulted in the system occasionally taking high resolution images when it was not necessary. This issue could be solved in future work by having the system update the composite background image more often, or by having it further reduce lighting noise by searching for periodic variations.

Because of the versatile nature of the high/low resolution camera combination, future work with this system is likely. Issues to be addressed include obtaining more computing power to allow for more complex algorithms that still run in real-time, and implementation of several security camera systems running together over a network. Additionally, this system could be evolved to a biometric identification system, which looks for faces and performs high resolution facial recognition. Finally, the algorithms could easily be expanded to look for suspicious events besides abandoned luggage, such as the presence of weapons, people running, or fights.

# 9. References

[1] C. Wren, A. Azabayejani, T. Darrel, and A. Pentland, “Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, pp. 780-785, July 1997.

[2] N. Friedman and S. Russell, “Image segmentation in video sequences: A probabilistic approach," *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence* (UAI-97), pp. 175-181, Morgan Kaufmann Publishers, Inc., (San Francisco, CA), 1997.

[3] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62-66.

[4] Performance Evaluation of Tracking and Surveillance Conference 2006, “PETS 2006 Benchmark Data”, http://www.cvg.rdg.ac.uk/PETS2006/data.html.

[5] Gonzalez, Rafael, Richard Woods and Steven Eddins, 2004. *Digital Image Processing Using MATLAB*. Upper Saddle River, NJ: Pearson Prentice Hall.

# 10. Works Consulted


1. Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from videos," *Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.

2. L. Li, W. Huang, I. Yu-Hua Gu, and Q. Tian, " Statistical Modeling of Complex Backgrounds for Foreground Object Detection," *IEEE Transactions On Image Processing*, Vol. 13, No. 11, pp. 1459-1472, 2004.

3. J. Papon, R. Broussard and R.W. Ives, "Dual Camera System for Acquisition of High Resolution Images," *Proceedings of the SPIE,* Vol. 6496, pp. 29-37, 2007.

4. B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge: Cambridge University Press, 1996.

5. T. Masters, Signal and Image Processing with Neural Networks, New York: Wiley, 1994.

6. R. Duda, P. Hart and D. Stork. Pattern Classification 2nd Ed., New York: John Wiley and Sons, 2001.

7. Tekalp, Murat. Digital Video Processing, Upper Saddle River, NJ: Prentice Hall, 1995.

8. R. Jain, R. Kasturi, and B. Schunck. Machine Vision, Boston: McGraw Hill, 1997.

9. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," *Proceedings of SPIE Visual Communications and Image Processing*, Vol. 5308, pp. 881-892, 2004.

10. L. Tian, M. Lu, and A. Hampapur, "Robust and efficient foreground analysis for real-time video surveillance," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 1182 – 1187, 2005.

# Appendix A: Glossary

**Complement –** A binary operation which replaces all "true" values with "false" values and vice-versa.

**Dilation –** A morphological operation which expands the size of "true" regions.

**Graphical User Interface (GUI) –** GUI's are interfaces which contain graphical elements for easy end-user control of a computer algorithm or system.

**Majority Filter –** A type of filter that processes pixels in a region of an image and sets the center pixel of the region to the dominant value.

**Mask –** A type of binary image consisting of only logical values of true or false. In image processing, masks are commonly used to eliminate unwanted areas from images (for example, regions with false values are removed).

**MATLAB** - a numerical computing environment and programming language created by The MathWorks.

**Mega-Pixel-** A mega-pixel is 1 million pixels, and is a term used not only for the number of pixels in an image, but also to express the number of sensor elements of digital cameras.

**MEX-File –** MEX stands for MATLAB executable. MEX-files are dynamically linked subroutines produced from C/C++ or Fortran source code that, when compiled, can be run from within MATLAB in the same way as MATLAB files or built-in functions.

**Morphology - A** collection of techniques for digital image processing based on mathematical morphology. Since these techniques rely only on the relative ordering of pixel values, not on their numerical values, they are especially suited to the processing of binary images and grayscale images whose light transfer function is not known.

**Packet –** In digital communications, a packet is a formatted block of information transmitted and/or received on a computer network.

**RS-232 –** A standard for serial binary data commonly used in computer serial ports.

**Slew Rate –** A measurement used for mechanical devices which move in a circle. It refers to the number of degrees the device turns in one second.

**Software Development Kit (SDK) –** A set of development tools that allows a software engineer to create applications for a certain software package, software framework, or hardware platform.

**Universal Serial Bus (USB) -** A serial bus standard used to connect hardware devices to a computer.

# Appendix B: Graphical User Interface

The MATLAB Graphical User Interface (GUI) was created specifically for this research, and allows the user to control all aspects of the system. This appendix details the various functions of the GUI.

**Initialize Button –** This button initializes the controls for all various parts of the subsystem. It sets up the serial port connection to the pan/tilt mount, opens a USB connection and turns on the high resolution camera, and opens a USB connection and turns on the webcam. If any of these components are not present, it will notify the user of an error.

**Clean Up Button –** This button disconnects the various components of the system (in software). It can be used if an error occurs to reinitialize the system, and is done automatically when the program exits.

**Pan/Tilt Configuration / Control**

*Max Speed* – This defines the maximum speed the pan/tilt mount will attain as it moves. It is defined in degrees/second, and can be set for values from 1-400.

*Acceleration* – This defines the maximum acceleration rate the pan/tilt mount will achieve as it moves. It can be set for values from 0.1 to 200 degrees per second squared. Note: Setting this to a value of over 130 causes significant jarring of the high resolution camera, especially for small movements, and is not recommended.

*Base Speed* – This defines the base speed of the pan/tilt mount, and should be set to a value of 10 degrees/second. Anything higher than this will cause significant jarring on small movements and could damage the high resolution camera.

*Query Speeds* – This queries the pan/tilt controller for the different speeds listed above, and fills the text boxes with the appropriate values.

*Set Speeds* – This sends the speeds shown in the text boxes above it to the pan/tilt controller. It will issue a warning if dangerous values are entered, and an error if invalid ones are entered.

*Pan/Tilt Motor Response* – This dialog box shows all the commands sent to the pan/tilt controller, and its machine language responses. It can be used to make sure that any commands sent to the mount have been successfully received.

*Pan/Tilt Manual Control* – These dialog boxes allow the user to manually control the pan and tilt of the mount. Valid values for pan (left right) are from -2800 to 3000, and from 1500 to -1500 for tilt (up/down).

**Canon Camera Configuration / Control**

*Image Size* – This defines the size of the image taken by the high resolution camera. The three available options are Large (2800 x 2400), Medium (1920x1680) and Small (640x480).

*Auto Shutter Release* – This allows the algorithms to automatically move and capture images of abandoned luggage. As a precaution, ensure that nothing is touching the pan/tilt mount when this option is enabled, and that there are no wires that might get caught when the mount moves (It can easily cut through them if the acceleration and maximum speed values are set on the high end).

*Shutter Release* – This manually releases the high resolution camera shutter, and downloads the image acquired to the GUI.

*Zoom* – This is a manual control for the high resolution camera zoom, and should not be tampered with while Auto Shutter Release is enabled.

*Show Last Still Shot* – This displays the last image acquired by the high resolution camera in a window in the GUI.

**Open Webcam Preview Button –** This opens a window which shows a live video feed from the webcam. This button should be pressed before Auto Shutter Release is enabled.

**Start Difference Image Button –** This button starts the foreground segmentation and background tracking algorithms. Additionally, it opens a window which shows a visual representation of what the tracking algorithm is detecting.

# Appendix C: MATLAB Code

The following appendix contains listings of the MATLAB code used in the final algorithms. Code for the Graphical User Interface is not included for brevity's sake, but can be obtained by contacting the author.

**Buildbackgroundgraymedian.m –** This is the function to create the composite background image.

```
%Composite Background Creation
%Jeremie Papon, 2007
function backimg = buildbackgroundgraymedian(file)

    %Open the first video frame to get size information about it and
    %allocate the appropriate amount of memory
    pic1 = imread(file(1).name);
    backimg = uint8(zeros(size(pic2,1),size(pic2,2)));
    picarray = zeros(size(pic2,1),size(pic2,2),330,'int16');

    for picindex=1:100
        %Open the video frames to analyze them
        pic1 = cvcolor_mex(imread(file(picindex).name),'rgb2gray');
        pic2 = cvcolor_mex(imread(file(picindex+1).name),'rgb2gray');
        %Calculate the Difference Mask
        diffimg = imabsdiff(pic1,pic2);
        %Threshold the Difference Mask
        diffimg = graythresh(diffimg);
        %Do a majority Operation to reduce noise from temporal differences
        diffimg = bwmorph(diffimg,'majority');
        %Dilate the Image to increase areas of movement
        diffimg = cvlib_mex('dilate',diffimg,6);
        %Multiply by -2 so all moving areas are negative
        tempimg = int16(diffimg .* (-2));
        %Add one so movement in -1 and movement is 1
        tempimg = tempimg + 1;
        %Multiply Difference mask by video frame
        temp = int16(int16(pic1) .* tempimg);
        %Add the result into the motionless structure
        picarray(:,:,picarrayindex) = int16(temp(:,:));


    end


%Cycle through each (x,y) pixel location
    for x = 1: size(pic2,1)
```

```
        for y = 1: size(pic2,2)
            %Find all values for this location that were motionless
            tempvals = picarray(x,y,find(picarray(x,y,:)>=0));
            %Set the background value for this location to the median of
            %all motionless values for this location
            if(size(tempvals)>0)
                backimg(x,y) = uint8(median(tempvals));
            else
                backimg(x,y) = 0;
            end
        end
    end
```

**Subwaydemo.m** – This file shows a demo of the foreground segmentation and object tracking algorithms.

```
%Foreground Segmentation and Object Tracking Demo
%Created by Jeremie Papon, 2007

%Initialize Tracking Structure
objectlist =
struct('XbyY',{},'WbyH',{},'idleframes',{},'speed',{},'color',{});
%Set Various Thresholds
SPEEDTHR = 2.0;
DISTTHR = 40;
SIZETHR = 30;
%Initialize Morphological Mask
SE = strel('square',10);

%Open the PETS2006 Data Set and Read it into memory
    dos('dir /B c:\papon\SubwayData\s7-t6-b\video\pets2006\s7-t6-b\3\*.jpeg >
filelist.txt');
    fid=fopen('filelist.txt');
    index=1;
        while ~feof(fid)
          file(index).name = ['c:\papon\SubwayData\s7-t6-b\video\pets2006\s7-
t6-b\3\' fgetl(fid)];
          index=index+1;
        end
    fclose(fid);
    index = index -1;

%Call Composite Background Creation Function
backim = buildbackgroundgraymedian(file);
background = backim;
background = imresize(background,.5);
    %Read in first frame, Convert to Grayscale, and Reduce it in Size
    newframecolor = imread(file(1).name);
    newframe = cvcolor_mex(newframecolor,'rgb2gray');
    newframe = imresize(newframe,.5);
    %Create the "old" background difference mask
    oldbackdiff = imabsdiff(background,newframe)>20;
```

```
    for n = 2: size(file)
        %Clear the Deletion Array for Tracked Objects
        deletearray = logical(ones(size(objectlist)));
        %Read in the next frame, convert to grayscale, and resize it
        newframecolor = imread(file(n).name);
        newframe = cvcolor_mex(newframecolor,'rgb2gray');
        newframe = imresize(newframe,.5);
        %Create the "new" background difference mask
        newbackdiff = imabsdiff(background,newframe)>20;
        %Determine where the two background differences are both 1 (and AND)
        backdiff = oldbackdiff + newbackdiff > 1;
        %Set the "old" background difference to the new one
        oldbackdiff = newbackdiff;
        %Perform a Majority Operation on the difference mask
        backdiff = bwmorph(backdiff,'majority');
        %Perform the morphological closing operation on the difference mask
        backdiff = imclose(backdiff,SE);

        %Create a label matrix for the foreground mask
        labelmat = bwlabel(backdiff,4);
        %Find the contiguous regions in the mask
        stats= regionprops(labelmat,'BoundingBox');

        %This Loop Cycles through all of the found objects
        for boxes = 1: size(stats,1)
            %Read location and size information
            Xloc = stats(boxes).BoundingBox(1);
            Yloc = stats(boxes).BoundingBox(2);
            Width = stats(boxes).BoundingBox(3);
            Height = stats(boxes).BoundingBox(4);
            newobject = 1;
            %This cycles through all old objects
            for index = 1:size(objectlist,2)
                Xdist = abs(Xloc - objectlist{index}.XbyY(1));
                Ydist = abs(Yloc - objectlist{index}.XbyY(2));
                Widthdiff = abs(Width - objectlist{index}.WbyH(1));
                Heightdiff = abs(Height - objectlist{index}.WbyH(2));
                %Find if the current object is the same as an old object
                if((Xdist + Ydist) < DISTTHR && (Widthdiff + Heightdiff <
SIZETHR))
                    objectlist{index}.WbyH = [Width Height];
                    objectlist{index}.XbyY = [Xloc Yloc];
                    objectlist{index}.speed =(objectlist{index}.speed+
sqrt((Xdist)^2 + (Ydist)^2))/2;
                    %If it is and old object, Check to see if Idle
                    if(objectlist{index}.speed < SPEEDTHR)
                        objectlist{index}.idleframes =
objectlist{index}.idleframes + 1;
                        if(objectlist{index}.idleframes > 15 &&
objectlist{index}.idleframes < 25)
                            objectlist{index}.color = 'y';
                        elseif(objectlist{index}.idleframes > 24)
                            objectlist{index}.color = 'r';
                        end
                    else
```

```
                        objectlist{index}.idleframes = 0;
                        objectlist{index}.color = 'g';
                    end
                    %Dont Delete This Object
                    deletearray(index) = 0;
                    %Not a new Object
                    newobject = 0;
                    break;
                end
            end
            %Create a New Object if no old objects matched this one
            if(newobject)
                newobj = size(objectlist,2)+1;
                objectlist{newobj}.XbyY = [Xloc Yloc];
                objectlist{newobj}.WbyH = [Width Height];
                objectlist{newobj}.idleframes = 0;
                objectlist{newobj}.speed = 0.1;
                objectlist{newobj}.color = 'g';
                deletearray(newobj) = 0;
            end
        end
        %Delete all objects from the object list that were not scene this
        %iteration
        objectlist(deletearray) = [];

        %This Creates the four paned movie output showing the various parts
        %of the algorithms
         movieimg = uint8(zeros(size(background,1)*2,size(background,2)*2));
         movieimg(1:size(background,1),1:size(background,2)) = background;

movieimg(1:size(background,1),size(background,2)+1:size(background,2)*2) =
uint8(oldbackdiff)*255;

movieimg(size(background,1)+1:size(background,1)*2,size(background,2)+1:size(
background,2)*2) = uint8(backdiff)*255;

movieimg(size(background,1)+1:size(background,1)*2,1:size(background,2)) =
newframe;
         imshow(movieimg);
          for index = 1:size(objectlist,2)

rectangle('Position',[objectlist{index}.XbyY(1),objectlist{index}.XbyY(2),obj
ectlist{index}.WbyH(1),objectlist{index}.WbyH(2)],'FaceColor',objectlist{inde
x}.color,'EdgeColor',objectlist{index}.color)

text(objectlist{index}.XbyY(1)+floor(objectlist{index}.WbyH(1)/3),objectlist{
index}.XbyY(2)+floor(objectlist{index}.WbyH(2)/3),num2str(objectlist{index}.i
dleframes))
          end

    end
```

# Appendix D: Papers Published

Two papers were published based on this research. The first was published in the Proceedings of the 19th Annual SPIE/IS&T Symposium on Electronic Imaging. The second was published in the Proceedings of the 21st National Conference on Undergraduate Research. Both papers have been included here.

# Dual Camera System for Acquisition of High Resolution Images

Jeremie A. Papon, Randy P. Broussard, Robert W. Ives
United States Naval Academy, Annapolis, MD 21412

## ABSTRACT

Video surveillance is ubiquitous in modern society, but surveillance cameras are severely limited in utility by their low resolution. With this in mind, we have developed a system that can autonomously take high resolution still frame images of moving objects. In order to do this, we combine a low resolution video camera and a high resolution still frame camera mounted on a pan/tilt mount. In order to determine what should be photographed (objects of interest), we employ a hierarchical method which first separates foreground from background using a temporal-based median filtering technique. We then use a feed-forward neural network classifier on the foreground regions to determine whether the regions contain the objects of interest. This is done over several frames, and a motion vector is deduced for the object. The pan/tilt mount then focuses the high resolution camera on the next predicted location of the object, and an image is acquired. All components are controlled through a single MATLAB graphical user interface (GUI). The final system we present will be able to detect multiple moving objects simultaneously, track them, and acquire high resolution images of them. Results will demonstrate performance tracking and imaging varying numbers of objects moving at different speeds.

Keywords: surveillance, real-time video, biometrics, MATLAB.

## 1 INTRODUCTION

Currently several cities and airports have implemented large scale security camera networks which seek to use stationary cameras for security. The primary problem with these CCTV systems is the quality of the images they take; security cameras are generally of low resolution, which greatly restricts the accuracy of any processing that might be done on the images. This project seeks to resolve this problem by developing a dual camera system which contains a low resolution web camera and a high resolution still-frame digital camera. The low resolution camera remains stationary, while the high resolution camera is mounted on a motorized pan/tilt mount. The low resolution video is used for monitoring and detecting objects of interest using a MATLAB graphical user interface (GUI). This same GUI controls the pan/tilt mount, which points the high resolution camera and then controls the capture of the still-frame images.

## 2 DESCRIPTION OF COMPONENTS

### 2.1 Low resolution camera

The low resolution camera is a Logitech Quickcam Pro 5000 (Figure 1). This camera has proven ideal for the purpose of surveillance because it combines good resolution (640 x 480 pixels), fast frame rates (30 frames per second), and a wide field of view (82 degrees) in a small robust package. The Quickcam is stationary, and surveys an entire area of interest at once, continuously monitoring for movement. When movement in the field of view is detected, the camera output is processed for object localization. The Quickcam interfaces directly with MATLAB via a USB cable, which allows for quick processing of the video feed.

### 2.2 High resolution camera

The high resolution camera consists of a 6 mega-pixel Canon Powershot S3IS (Figure 2). This camera was chosen because of its good low light performance and motorized 12 times optical zoom. The optical zoom is extremely fast, and can be controlled remotely, which allows for imaging of small objects. The camera is mounted on a high speed

pan/tilt mount for pointing, and comes with a Software Development Kit (SDK) which allows for remote control of the camera over a USB cable, including zoom and capture functions.

**Figure 1: Quickcam Pro 5000**

### 2.3 Pan/Tilt mount

The pan/tilt mount is a Directed Perception PTU-D46-17 which can pan a full 360 degrees and tilt 180 degrees, at speeds of over 300 degrees a second. The mount allows for fast, stable, and accurate pointing of the high resolution camera. The pan/tilt mount is controlled serially, and can be commanded directly from the GUI using the serial port functionality built into MATLAB.

## 3 INTERFACING OF COMPONENTS

The first major hurdle in developing a system that combines various off-the-shelf components into a single working unit is interfacing. Both the high resolution camera and the mount required separate controls to be developed in order to meet their desired functionality. This was accomplished in MATLAB.

Control of the Pan/Tilt mount (Figure 3) was accomplished using the computer's serial port. The serial port is used to send commands to the Pan/Tilt mount's controller, and allows control of both the movement of the mount and all of its settings, such as slew rate and acceleration.


**Figure 2: Canon Powershot S3IS**

Figure 3: Directed Perception Pan/Tilt Mount

Controlling the Canon Powershot proved to be much more difficult, primarily because the Canon SDK, which provides a basic framework for communicating with their cameras, was written in the C programming language. In order to use the camera, all of the control functions had to be ported into MATLAB using MEX files. MEX files are a specific type of MATLAB file which are written in C, but can compile and run in MATLAB. MEX files for initializing the camera, changing camera settings, and controlling the shutter were created. But the actual transfer of images from the high resolution camera to the computer over the USB (Universal Serial Bus) cable proved problematic, as the Canon framework for doing this could not be ported into MATLAB. As such, new functions were written in C which transferred the image data from the camera over the USB cable in individual packets. Packet sizes in initial testing were limited to only 1024 bytes, which resulted in very poor transfer speeds for images (upwards of 5 seconds). After several adjustments and revisions to the transfer protocol, 128 kilobyte packets were used successfully, which increased transfer speeds substantially (approximately one second for a one megabyte image). The complete assembled system can be seen in Figure 4.

Figure 4: The System

# 4 OBJECT DETECTION

## 4.1 Foreground extraction

The first step in detecting motion in video is extracting regions of interest from the background. Two methods were implemented to do this; motion segmentation and background subtraction. Both are able to extract the foreground, but differ somewhat in speed and accuracy.

### 4.1.1 Motion segmentation

Motion segmentation is implemented by finding the absolute difference between the current frame and a previous frame and then thresholding the result. The equation for doing this is shown in equation (1) where pixel $x_k(m,n,c)$ in frame $k$ is subtracted from a frame $l$ frames ago, and thresholded using a threshold value $T$. Here, $m$ and $n$ represent the row and column numbers, and $c$ represents the three color planes of red, green, and blue.

$$\sum_{c=1}^{3} \left| x_k(m,n,c) - x_{k-l}(m,n,c) \right| > T \qquad (1)$$

After thresholding, morphological techniques are used. The morphology consists of erosion to eliminate noise, and then dilation to expand the areas of motion. This is done because it was determined in testing that it is better to get extra area surrounding detected motion in this step, rather than risk losing areas due to processing tolerances.

While motion segmentation is computationally efficient and very adaptable to changing backgrounds, testing has shown that it has limitations. Variations in lighting, video noise, and stationary foreground objects can cause it significant problems.

### 4.1.2 Background subtraction

Background subtraction, while more computationally intensive, can be more reliable. This technique requires building a composite background, which can be done in many different ways, each which have their own advantages and disadvantages. The most common methods are Kalman filtering[1], mixture of Gaussians[2], and median filtering[3]. For the purposes of this research, none of these techniques were computationally efficient enough to be applied, so a new technique was developed that builds on these techniques. The base premise of the approach is that pixel values for background will reoccur quite often over time, with slight variations due to lighting. As such, a composite background is built where weights are assigned for the values of each pixel, depending on frequency of occurrence over time. These weights are adjusted in each frame according to equation (2), where $w_i$ specifies the weight for intensity value $i$, $L$ is the learning rate, and $K$ is 1 for the new value and 0 for others.

$$w_i = w_i + \frac{1}{L}(K - w_i) \qquad (2)$$

As new values appear, they are added to the list of weights, and so over time, the most commonly occurring values for each pixel will be given large weights. If a foreground object moves into the scene, its weight value will be extremely low, and thus can be segmented.

## 4.2 Color segmentation

Once foreground regions are segmented, these regions are put through further processing to determine where color values corresponding to the objects of interest are located. Detection is based on the principle that in certain color planes, these values occupy a very narrow range. After extensive testing, the LUV color space was chosen because it allowed for the most accurate segmentation, while still using integer values (which allows for a fast processing technique that will be shown later). A pixel in the LUV color plane contains three values; one for luminance (brightness), and two for chrominance (color).

In order to implement the color segmentation in a manner as efficient as possible, the number of comparisons to determine if a pixel is, for example, skin must be kept as low as possible. This is necessary because every video frame contains 640x480 pixels, or 307,200 pixels. A brute force implementation would require 8 comparisons per pixel, as one needs to determine whether each of the three components of the color plane fall between a low and high threshold (that is, whether a pixel's color is "close" to that of an object of interest), and then two "AND" operations on these three binary values. These 8 comparisons per pixel amount to 2,457,600 comparisons per frame. This proved to be too slow, since 30 frames per second were being processed, so a new faster approach was implemented. In this approach, the range of values are placed in three data structures; one for each portion of the color space. A graphical representation of these structures is seen in Figure 5. The structures contain a logical 1 or 0 for every possible L, U, and V value (integers with value 0 to 255). These structures allow for much faster detections, as it only requires three quick "AND" operations to determine if a pixel lies within the range of color values. This reduces the maximum number of computations per frame from 2,457,600 to 921,600. In trials, this amounted to a reduction of about .05 seconds per frame (a significant amount of processing time saved for real time video).

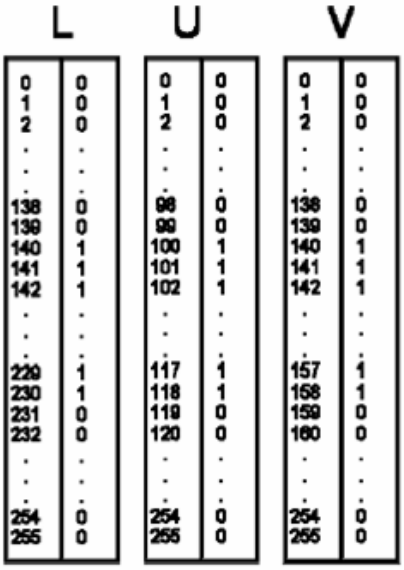


**Figure 5: Color Segmentation Data Structure**

### 4.3 Neural network for classification

At this point the video frame has been reduced to a few windows which may contain either the objects of interest. Once these windows are extracted, they must be classified as either objects of interest or non-objects of interest. To do this, a multilayer perceptron neural network classifier is employed. Neural networks, named after their resemblance in functionality to the human nervous system, consist of large numbers of simple processing elements called neurons, which exhibit complex global behavior based on their connections and individual neuron parameters. Neural networks are used primarily to solve problems for which no mathematically optimal solution exists; such as determining whether or not an image contains a type of object.

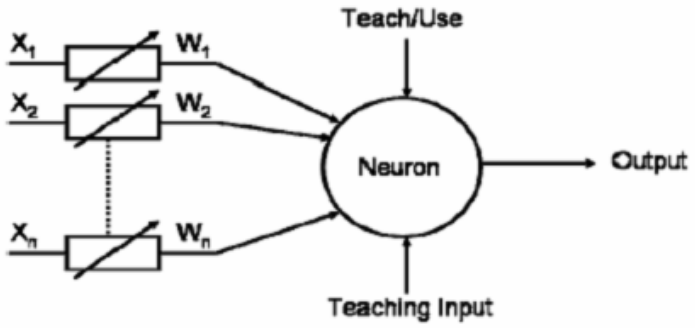


Figure 6: A Neuron

Each individual neuron in a neural network is simple in nature; it consists of a summation and a thresholding function. An example of a neuron is shown in Figure 6. Each input $X_i$ is multiplied by a weight $W_i$ and these products are summed in the neuron. This value is then thresholded by a sigmoid function, and the output, either a one or zero, is sent to its output, which may be the next layer of nodes. The type of neural network employed in this research is called a multilayer perceptron feed-forward network[4]. This type of neural network is characterized by two principal elements; all of the inputs are connected to every hidden node, and information only goes forward (i.e. there is no feedback). This type of network was chosen because of its speed; since there is no feedback, all computations can occur simultaneously, allowing very fast classification. An example of a feed forward network similar to the one used is displayed in Figure 7.

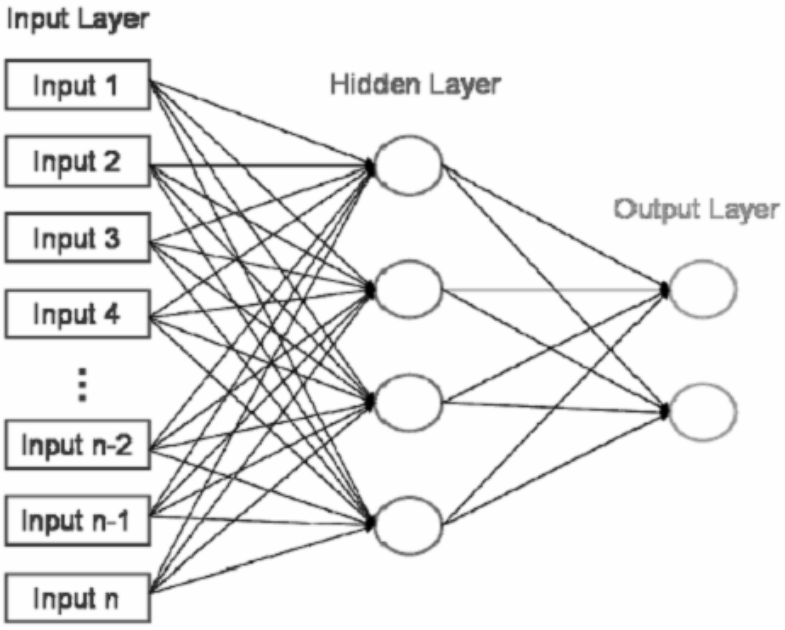


Figure 7: A Feed Forward Neural Network

Figure 7 shows a neural network that has n input values presented at a time to each of four neurons in a "hidden" layer. The output of these hidden neurons is passed to each of two neurons in the output layer. As discussed earlier, each neuron or "node" has weights which determine the output for any given set of inputs. In order to determine the values of the weights, and thus get the correct output, the network must be trained. This is accomplished by building a set of training data, which contains thousands of examples of objects of interest and non interest. Objects of interest are simply images of what we are trying to find. Objects of non-interest on the other hand can be quite literally, anything. As such, for the purpose of building this training data, random windows of images which did not contain objects of interest were used. Finally, in order for the windows to be processed by the network code, they must be changed from image matrices to one-dimensional vectors.

Once each window is classified (as of interest or non-interest) and transformed into a vector, a matrix is created in which each row represents a window's vector, with the first column containing the correct classification. This matrix is then used iteratively to find the best weights for the hidden nodes so that the neural network obtains the best classification performance possible (since the correct classification is known during training). One of the key variables in training is the number of hidden nodes; various numbers of nodes were tested in order to determine the number of hidden nodes that gave the best accuracy. Six hidden nodes worked the best, giving fast computation times and over 96% accuracy on the training set.

One of the other key variables is the number of input values passed into the neural network for each classification (that is, number of elements in each input vector). For this research, this amounted to the number of pixels used for a window. Various window sizes were tested, ranging from as small as 6x6 up to 30x30. It was found that increasing resolution past 12x8 actually resulted in a decrease in performance (both in efficiency and accuracy). While at first this may seem counter intuitive, it is a common occurrence in neural networks. It is primarily due to the fact that objects only have so many distinguishing features common to all of them. Once the resolution is increased beyond a certain point, the extra information provided by the extra pixels is spurious at best, and often detrimental to performance.

## 5 OVERALL OPERATION

Once a window is classified (manually) as an object of interest, pertinent information (centroid location, size, and speed) about the window is stored in a data structure. Only location and size are stored in the initial detection, with speed being stored after the same object is detected in four successive frames. Speed is determined by a weighted average of differences between the centroids from successive frames. The equation for doing this is shown in equation (3), where $n$=1 is the most recent frame, and $k_n$ are the weight average constants.

$$\Delta x = \sum_{n=1}^{3} k_n (|x_n - x_{n+1}|)$$
$$\Delta y = \sum_{n=1}^{3} k_n (|y_n - y_{n+1}|) \quad (3)$$
$$k_n = [.5, .3, .2]$$

The weighted average allows for changes in speed, while preventing excessive spikes due to errors in window centering. Once a speed is determined, the object is queued for capture. When the object reaches the beginning of the queue, the pan/tilt mount is moved to its next location (as predicted by the object's speed), and a high resolution image is taken. This method is particularly robust to error because a window is only queued for imaging if it occurs four times in a row. For something other then an object of interest to be queued, it would have to be falsely detected in four successive frames, which testing has shown is extremely unlikely. The entire process is run from one MATLAB graphical user interface, which gives options for all possible settings for the mount and cameras. The GUI also allows for manual capture of areas in the low resolution video frame.

## 6 RESULTS

The system has been undergone some preliminary testing that has proven its effectiveness for high resolution surveillance. The system allows for almost any type of object recognition algorithm that incorporates video processing to be inserted as the driving force for the high resolution camera. The system is virtually independent of the recognition algorithm used, and the end product is a high resolution image of objects of interest. Performance is dependent on a number of factors, primarily how well the recognition algorithm employed is able to distinguish objects of interest, but also the number and speed of the objects, to ensure they are not moving too fast for the servo motor to drive the high resolution camera to capture an image.

## 7 CONCLUSIONS

Biometric identification, public surveillance, and unmanned vehicle systems all currently suffer from one common handicap; their use of low resolution video. This severely restricts their ability to accurately achieve their respective goals. The conclusion of this research is a functioning system that addresses this problem by allowing the accurate and efficient capture of high resolution images of objects. The ability to detect any type of predefined object and autonomous nature of the end-system allows for simple application to all three of these areas, and more.

## 8 ACKNOWLEDGEMENTS

Author # 1 gratefully acknowledges the Office of Naval Research for partial support of this work, via the Naval Academy Trident Scholar Program, on funding document N0001406WR20137.

## 9 REFERENCES

1. C. Wren, A. Azabayejani, T. Darrel, and A. Pentland, "Pfinder: Real-time tracking of the human body," IEEE Transactions on Pattern Analysis and Machine Intelligence 19, pp. 780-785, 1997.

2. N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence, **UAI-97**, pp. 175-181, 1997.

3. Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from videos," Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance, 2001.

4. S. Haykin, *Neural Networks: A Comprehensive Foundation,* pp. 138-229, Macmillan College Publishing Co.: New York, 1994.

Proceedings of The National Conference
On Undergraduate Research (NCUR) 2007
Dominican University of California
San Rafael, California
April 12 – 14, 2007

# Autonomous Detection and Imaging of Abandoned Luggage in Real World Environments

Jeremie Papon
Electrical Engineering Department
United States Naval Academy
105 Maryland Avenue
Annapolis, MD 21402-5025. USA

Faculty Advisors: R. Broussard, R.W. Ives

## Abstract

This research developed a system that is able to detect and produce high resolution imagery of unattended items in a crowded scene, such as an airport, using live video processing techniques. Video surveillance is commonplace in today's public areas, but as the number of cameras increases, so do the human resources required to monitor them. Additionally, current surveillance networks are restricted by the low resolution of their cameras. For example, while there is an extensive security camera network in the London Underground, its low resolution prevented it from being used to autonomously identify the terrorists that entered the train stations in July 2005. With this in mind, this project developed a surveillance system that is able to autonomously monitor a scene for suspicious events by combining a low resolution camera for surveillance (a webcam) with a moving high resolution camera (a 6 mega-pixel digital still-frame camera) to provide a greater level of detail. This enhanced capability is used to determine whether or not the event is a threat.

For the purposes of this research, suspicious events were defined as a person leaving a piece of luggage unattended for an extended period of time, for example, as a terrorist might do when placing a bomb. Initial analysis of the surveillance video involved separating the foreground (such as people carrying luggage) from the background. In order to do this using live video, an automated algorithm was developed which creates a composite background image from a small number of video frames. In the algorithm, areas detected as motion were subtracted out from individual frames. These processed frames, which represented regions of no motion, were then combined by taking the median value for each pixel across all frames. These median values were used to form a composite background image which contained only the non-moving parts of the scene (i.e., the background).

Once this background was obtained, the system then detected live motion. Using a variety of filtering techniques, individual foreground objects were separated from the stationary background. These objects were then tracked over time. When an object (for the purposes of this research, a moving object was assumed to be a person) divided into two different objects, they were then tracked to see if one of the objects remained motionless. In doing so, the system was able to detect an "abandonment" event. When such an event occurred, an event timer began to determine how long the luggage had been abandoned. If the luggage was left unattended for a preset amount of time, the system tagged it for high resolution imaging.

Once an abandoned item was tagged for high resolution imaging, the system used a motorized pan/tilt mount to point the high resolution camera and acquire a high resolution image of the item. This image was then sent to a human supervisor for further investigation. The final security system allows a single person to monitor a vast array of camera systems (spanning for example, an entire airport) for abandoned luggage or any other pre-defined suspicious event.

**Keywords: Surveillance, Live-Video, Motion-Detection**

## 1. Introduction

Video surveillance is extremely common in modern public areas, and as the number of cameras increases, so do the number of video feeds to watch. This places an increasing burden on security personnel, as more and more must be assigned to the task of watching the videos. Additionally, a person is only able to watch so many video feeds at the same time with any sort of efficiency, and as with any system involving human monitors, are subject to human error. Take for example the London Subway Bombings of July 7th 2005, in which the terrorists passed directly in front of several security cameras (Figure 1). Even though the terrorists were on several watch lists, and security personnel were in a heightened state of alert, they were allowed free access to the subways, and managed to leave their bags, which contained bombs, throughout the London Underground.


Figure 1: Terrorists in London Subway on July 7th 2005

In addition to the fallibility of human monitors, current surveillance networks suffer from another major drawback; their low resolution. This severely limits the ability to identify threats via automated algorithms, such as the one presented in this paper. The low resolution of current security systems also prevents the implementation of covert automated biometric identification of people. As such, this research has developed a low-cost dual camera system, which combines a low resolution video camera with a high resolution still-frame camera mounted on a moving pan/tilt mount. This allows for analysis of the video to determine objects of interest and automatic investigation of these objects using the high resolution camera.

## 2. Description of system components

The low resolution camera consists of a Logitech Quickcam Pro 5000 (Figure 2). This camera has proven ideal for the purpose of surveillance because it combines good resolution (640 x 480 pixels), fast frame rates (30 frames per second), and a wide field of view (82 degrees) in a small robust package. The Quickcam is stationary, and surveys the entire area of interest at once, continuously monitoring for movement. When movement in the field of view is detected, the camera output is processed for object localization. The Quickcam interfaces directly with MATLAB via a USB cable, which allows for quick processing of the video feed.

Figure 2: Quickcam Pro 5000


Figure 3: Canon Powershot S3IS

Figure 4: Pan/Tilt Mount

The high resolution camera consists of a 6 mega-pixel (a mega-pixel is one million pixels) Canon Powershot S3IS (Figure 2). This camera was chosen because of its good low light performance and motorized 12 times optical zoom. The optical zoom is extremely fast, and can be controlled remotely, which allows for imaging of small objects. The camera is mounted on a high speed pan/tilt mount for pointing, and comes with a Software Development Kit (SDK) which allows for remote control of the camera over a USB cable.

The pan/tilt mount is a Directed Perception PTU-D46-17 which can pan a full 360 degrees and tilt 180 degrees, at speeds of over 300 degrees a second. The mount allows for fast, stable, and accurate pointing of the high resolution camera. The pan/tilt mount is controlled serially, and can be commanded directly from the GUI using the serial port functionality built into MATLAB.

## 3. Methodology

The first part of reliable foreground segmentation is the creation of an accurate background image. This is easy in a laboratory, where one can simply clear the area of people and other foreground objects. Unfortunately, in real world applications it can be difficult, if not impossible, to clear an area in order to get a background image. Additionally, even if one clears an area and obtains an accurate background image when a camera is installed, any changes in the background (such as a light burning out, or furniture being moved) will wreak havoc on the system's foreground segmentation. With this in mind, this research has developed a robust algorithm (Figure 5) which is able to take live video and dynamically create a "composite" background image. Using this algorithm, backgrounds can be created even with people present in the field of view of the camera. Additionally, because the algorithm works quickly and can be used on live video, it can be used to periodically update the background image automatically to account for changes.

The first step in the background creation is simple frame to frame differencing and thresholding to detect regions that contain movement. Threshold values for creation of the binary image are calculated using Otsu's method [1]. This creates a binary difference image (or mask) where binary 1 represents motion. This difference mask is then processed with binary morphology (dilation) to account for noise in the thresholding and reduce the possibility of non-static areas passing through. This mask is then complemented so that static regions (those which will be called the background) are logical 1's in the mask. This mask is then multiplied by the current video frame to create a motionless image frame, which contains only background areas. This motionless image is then stored. This process is repeated until every pixel has at least 5 "motionless" values. In testing this was shown to take generally about 4 seconds (at 25 frames per second) for normal surveillance video clips. Once the array of motionless images is created the median value for each pixel is calculated, and used to form the final composite background image. An example of the background creation is shown in Figure 6. The surveillance clip used comes from a London train station, filmed for the "Performance Evaluation of Tracking and Surveillance Conference 2006" (PETS 2006) [2].

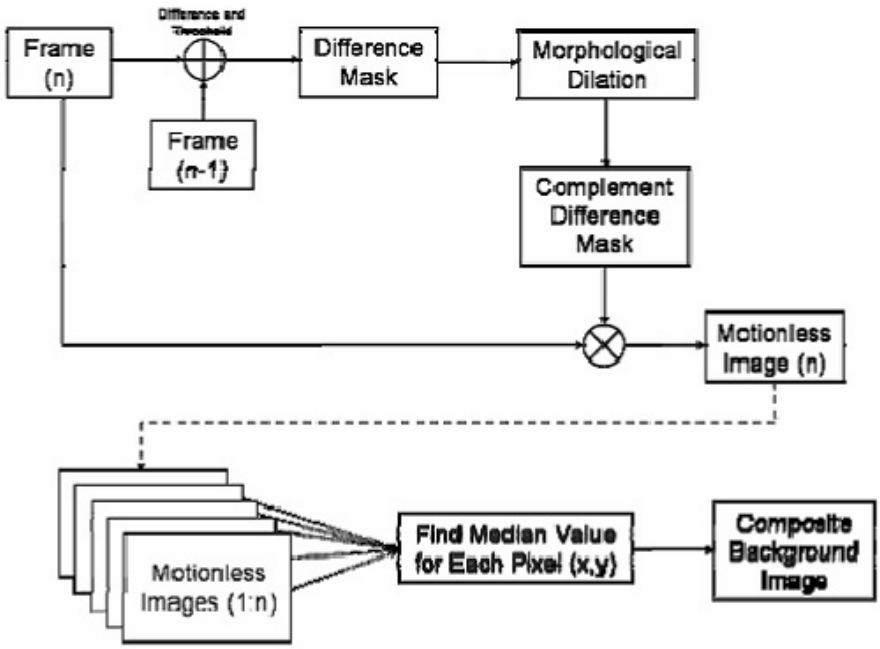


Figure 5: Composite background creation.

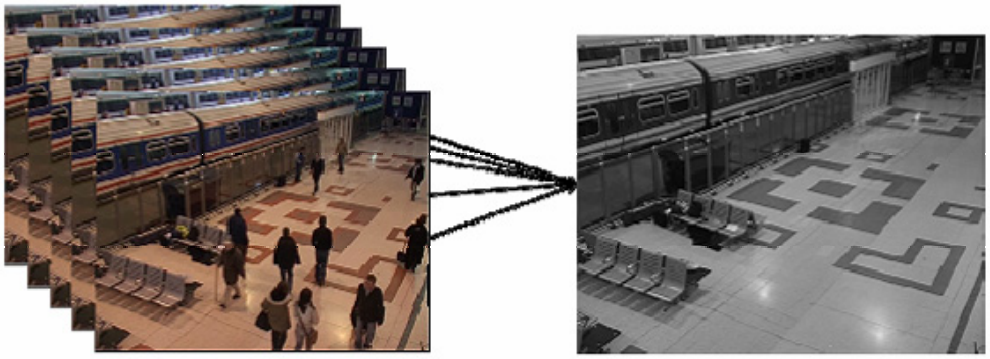

Figure 6: Example of composite background creation from live video

Once the composite background image is created, foreground segmentation is possible. This is accomplished using the algorithm shown in Figure 7. The first step in the process is differencing and thresholding both the current video frame and previous frame with the background image to create a binary mask, where binary 1 represents regions not in the background. These masks are then filtered using a 3 by 3 majority filter and then morphologically closed. These two different masks are then combined using a logical AND. This process of combining the current frame with the previous frame was shown to be extremely effective in testing, and resulted in more accurate segmentation results than could be attained by using only the current frame. An example of the foreground segmentation process applied to the PETS 2006 data set is shown in Figure 8. Once the foreground mask is obtained, the algorithm moves on to the next stage, object tracking.

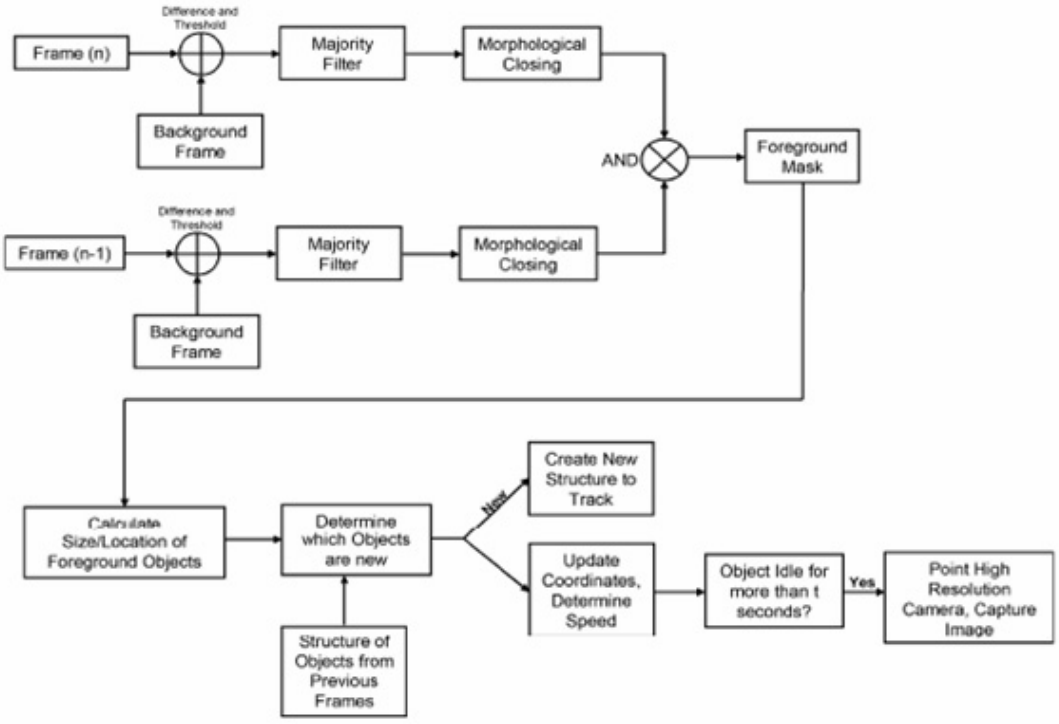


Figure 7: Foreground segmentation and object tracking algorithm

The final stage of the algorithm is the actual tracking of foreground objects. Foreground objects consist of regions of contiguous binary 1's in the foreground mask. The size and center of each of these regions is calculated, and compared with the objects present in the previous iteration. If an object is "new", it is stored in the object structure. In testing, new objects were either people entering the scene, a group of people separating, or a person and luggage separating. If an object was present in the previous frame, its statistics are updated and its motion vector (direction and speed) is calculated. This motion vector is used to determine if an object is idle; if the object remains idle for longer than a threshold time *t*, it is queued for imaging using the high resolution camera. In testing, it was shown that people standing still were rarely detected as "idle" because even if a person is not moving, they are seldom, if ever, perfectly still. Abandoned luggage on the other hand was detected with a great degree of reliability, because unlike people, it is perfectly motionless for significant amounts of time. When an object is queued for high resolution imaging, its size is used to determine zoom, and its coordinates are passed to the pan/tilt mount to point the camera.

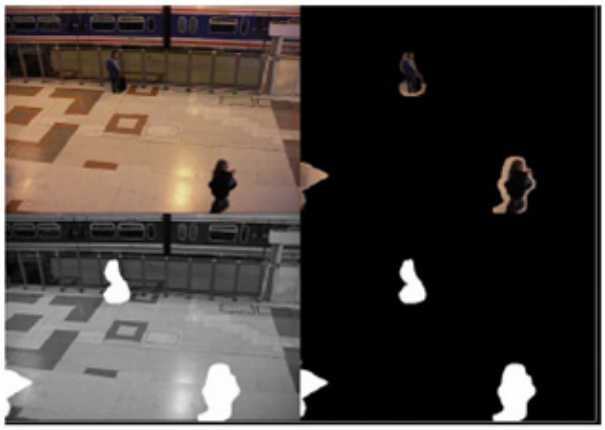

Figure 8: Example of foreground segmentation in London train station

## 4. Data

The first testing conducted was of the background creation algorithm. Five test scenario movie clips were used, 3 from the PETS2006 database and two from video of public areas recorded at the United States Naval Academy. Each of these test scenarios was processed 10 times, with the first test conducted starting at the beginning of the clip, the second at 10 seconds, the third at 20 seconds, and so on. The first test was used to determine the time it took to calculate the background pixel values from the motionless images using either an averaging or median. As can be seen in Figure 9, the median algorithm took longer in all five scenarios.

The next test focused on the accuracy of the background images created in the previous test. Because this research used pre-recorded clips, it was possible to obtain "perfect" background images; each clip contained at least one frame where there was no foreground present in the field of view. By using these as references, it was possible to test the accuracy of each background image created by taking the absolute error of each pixel in the composite backgrounds, and averaging them. This gives Figure 10, which shows the average error for the median and averaging algorithms in the five scenarios. It is readily apparent that the median algorithm is significantly more accurate in all cases.

The final stage of testing involved using the system in staged scenarios. The entire system was set up, and subjects were instructed to enact different scenarios. The tests ranged from a crowd of people walking back and forth, with no baggage abandoned to having everyone in the crow abandon luggage. As can be seen in Table 1, the system correctly detected abandoned luggage 100% of the time for all scenarios with fewer than 3 pieces of abandoned luggage (a drop consisted of a person abandoning an item they entered the scene with). Once it reached three drops, the system began to get saturated, and fail to detect abandoned items. False detections, which were defined as the system queuing a foreground object for photography when it was not abandoned luggage, were rare, especially in tests involving small numbers of people.

Table 1: results of testing on staged scenarios

| | | Correct Detection | Frames to Acquire Object as Idle | False Detections |
|---|---|---|---|---|
| **Nothing Idle, People Wandering** | | N/A | N/A | 1 |
| **1 Person, 1 Drop** | | **100%** | 12 | 0 |
| **Multiple People, 1 Drop** | | **100%** | 10 | 0 |
| **2 People 2 Drops** | | **100%** | | 0 |
| | *Bag 1* | TRUE | 20 | |
| | *Bag 2* | TRUE | 16 | |
| **Multiple People, 2 Drops** | | **100%** | | 1 |
| | *Bag 1* | TRUE | 24 | |
| | *Bag 2* | TRUE | 20 | |
| **3 People, 3 Drops** | | **100%** | | 0 |
| | *Bag 1* | TRUE | 20 | |
| | *Bag 2* | TRUE | 15 | |
| | *Bag 3* | TRUE | 14 | |
| **Multiple People, 3 Drops** | | **66.67%** | | 2 |
| | *Bag 1* | TRUE | 26 | |
| | *Bag 2* | TRUE | 22 | |
| | *Bag 3* | FALSE | N/A | |
| **Multiple People, 8 Drops** | | **75.00%** | | 1 |
| | *Bag 1* | TRUE | 28 | |
| | *Bag 2* | TRUE | 22 | |
| | *Bag 3* | TRUE | 24 | |
| | *Bag 4* | TRUE | 24 | |
| | *Bag 5* | FALSE | N/A | |
| | *Bag 6* | TRUE | 27 | |
| | *Bag 7* | TRUE | 21 | |
| | *Bag 8* | FALSE | N/A | |

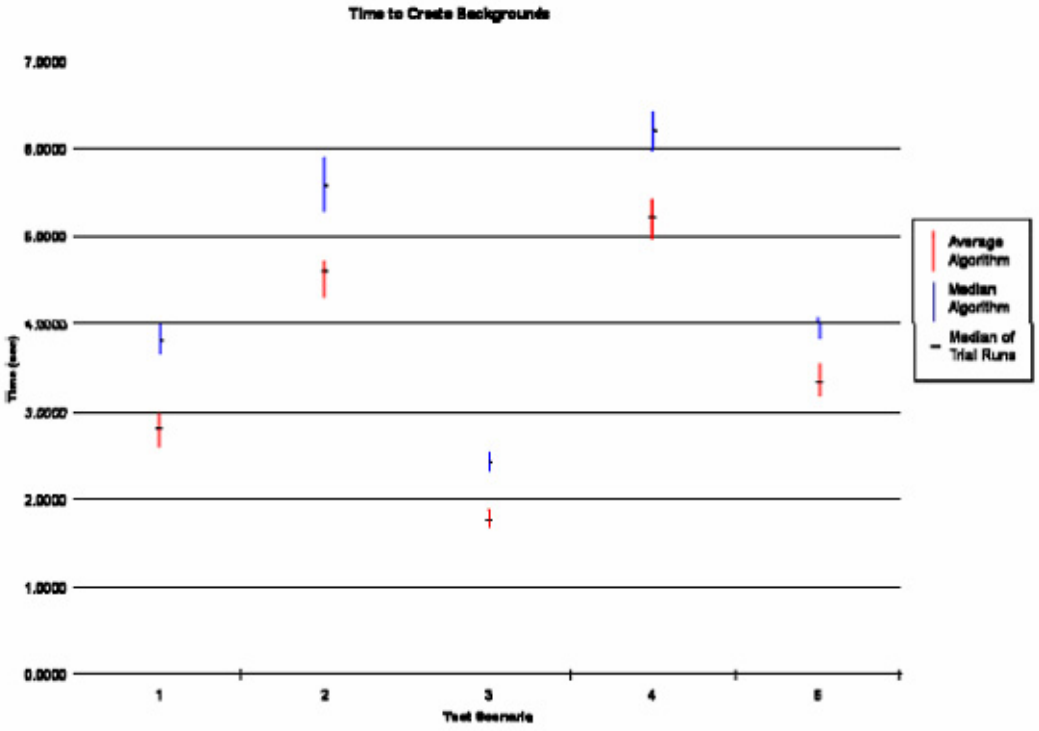


Figure 9: Time to create background images

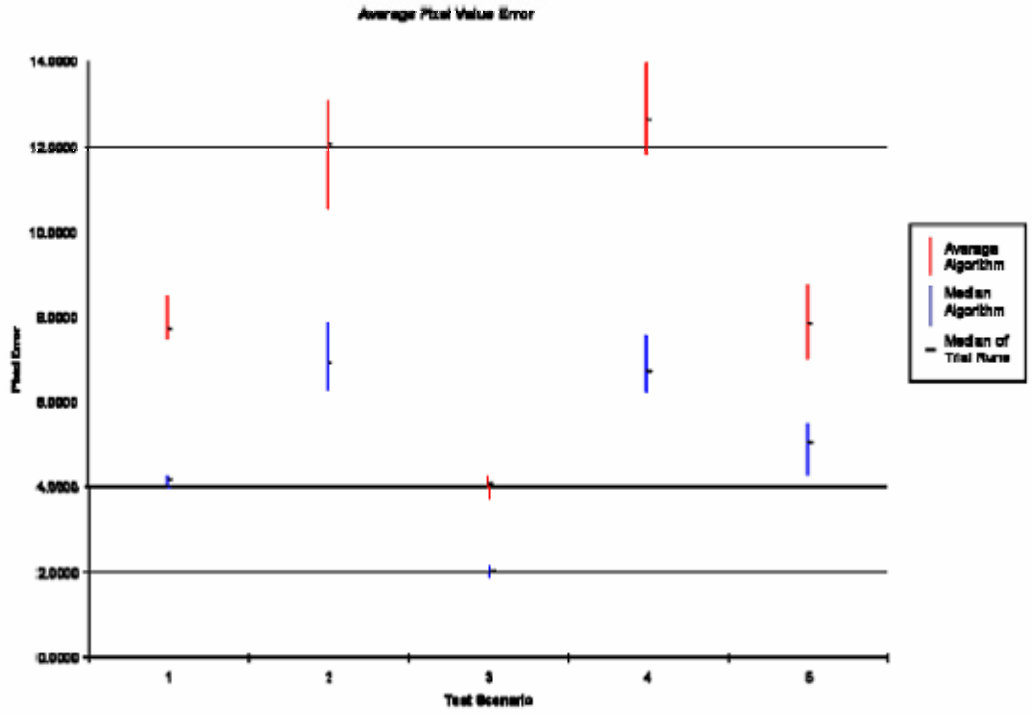


Figure 10. Error in background creation algorithms

## 5. Conclusion

One of the primary difficulties in working with an automated security system in the real world is that variations in the scenery can cause significant errors. Creating an algorithm that is able to adapt itself to any environment presented to it is one of the biggest hurdles to overcome in order to create a reliable autonomous security system. This research successfully accomplishes this with the background creation algorithm. The system's ability to rapidly create an accurate representation of an "empty" scene allows it to be deployed at any location and used immediately. Additionally, the use of temporal differencing in the foreground segmentation algorithm proved extremely successful for creating an accurate foreground mask.

The high resolution camera system proved to be extremely useful in testing, allowing for distant objects barely visible in the low resolution video to be photographed in great detail. While the objects could not be identified in the video feed, the high resolution photos allowed the human supervisor to positively identify them as abandoned luggage. A surveillance system could easily be set up using any number of the high/low resolution camera systems and monitored by a single supervisor, since they would only need to occasionally examine the high resolution photographed which the systems flagged as security risks.

The problems encountered in testing were primarily due to poor lighting conditions in the test area. Flickering and dim lights caused some error in foreground segmentation and also occasionally prevented the high resolution camera from focusing properly. The problem of flickering lights could be solved in future work by adjusting the segmentation to account for them (for example, adjusting the algorithm to ignore periodic noise). The problems with the high resolution imaging could be solved by using a true single-lens reflex (SLR) camera (none were readily available for use in this research), or a camera with better low light imaging capability.

## 6. Acknowledgements

The author wishes to express his appreciation to:
Dr. Robert Ives, Electrical Engineering Department, USNA – Primary Project Adviser
Dr. Randy Broussard, Weapons and Systems Engineering Department, USNA –Secondary Project Adviser
Mr. Jeffery Dunn, National Security Agency –External Collaborator
Ms. Daphi Jobe, Electrical Engineering Department, USNA- Electronics Technician
Mr. Jerry Ballman, Electrical Engineering Department, USNA- Laboratory Technician
Mr. Michael Wilson, Electrical Engineering Department, USNA – Laboratory Technician

## 7. References

[1] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[2] Performance Evaluation of Tracking and Surveillance Conference 2006, "PETS 2006 Benchmark Data", http://www.cvg.rdg.ac.uk/PETS2006/data.html

## 8. Works Consulted

1. N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence* (UAI-97), pp. 175-181, Morgan Kaufmann Publishers, Inc., (San Francisco, CA), 1997.

2. Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from videos," *Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.

3. L. Li, W. Huang, I. Yu-Hua Gu, and Q. Tian, " Statistical Modeling of Complex Backgrounds for Foreground Object Detection," *IEEE Transactions On Image Processing*, vol. 13, no. 11, Nov., pp. 1459-1472.

# Appendix E: Original Project Report

Due to unforeseen human subject testing issues, the nature of this Trident Project was modified in December 2006. While the hardware and its control functions for the original project were able to be used in the new project, the algorithms were not applicable to the new topic. As such, the original project report has been included here to document the accomplishments achieved during the first semester of the Trident project.

## Original Project Abstract

After the London subway bombings of 7 July 2005, British authorities were able to use recordings from the extensive surveillance system installed in the subway system to identify potential suspects, which proved a great boon to their investigation, and led to arrests 5 days after the bombings. Unfortunately, current surveillance camera networks only provide fuzzy low resolution imagery, and have no ability to conduct automated screening using facial recognition. Because of this, London's huge network of security cameras could not prevent the terrorist attacks; they could only help in the investigation.

The primary goal of this project is to develop a complete surveillance system that integrates a low cost, wide angle, low resolution camera with a high resolution digital camera in a complete self contained package. The system uses a high-speed motorized pan/tilt mount to allow the high resolution camera to focus in on any point in the low resolution camera's field of view. The low resolution camera is connected to a computer which runs a windows program that uses neural networks to locate individual faces in the field of view. These coordinates are then used to move the pan/tilt mount, so that the high resolution camera can take a close up image of the subject's face.

Testing involved the set up of one camera unit in a building of the Naval Academy, with volunteer Midshipmen used as subjects. The system ran continuously with data processing being done on a laptop. Individuals who volunteered were asked to walk down the hallway in varying numbers, from one to six people at a time. This allowed for controlled testing to determine the system's reliability and response speed.

# Table of Contents:

# Table of Figures

# 1. Biometrics and Security

Biometrics is the general term for a large field of study which consists of using digital signal processing to automatically recognize individuals based on intrinsic human traits. These traits can be anything which is unique about individuals; systems have been developed to recognize faces, irises, retinas, fingerprints, voice, and even gait. On the surface the problem may seem elementary; human beings instinctively recognize hundreds of people with few, if any, errors on a daily basis. The simplicity with which we recognize people disguises the difficulty of the problem. The study of biometrics over the past 15 years has shown that the problem is far more complex then it would seem at first. This fact does not surprise biologists, since over 50% of the brain is used for visual processing. In fact, contrary to what may seem logical, we use far more brain power in recognizing people than we do in purely intellectual pursuits such as mathematics [2].

The first issue to be dealt with is how to obtain standardized data. We, as humans, accomplish this instinctively; we simply turn our heads and look at the individual we wish to identify. Clearly, this task is far more complicated for a stationary camera to accomplish. Early biometric systems avoided this problem by only functioning effectively if the subject to be identified looked directly into a camera. Bypassing the problem by putting a constraint on where a subject must look does not provide a viable solution. There is a need for systems which are able to identify individuals without interacting with them. This leads to a more complex problem, which involves real time pattern recognition to locate individuals in a scene.

Secondly, the amount of data collected that must be processed in real time must be taken into consideration. Recorded signals of human traits, be they one-dimensional signals such as voice, or three-dimensional images of a person's face, contain far too much information to be processed directly. These images or recordings must first be analyzed using automated algorithms to reduce them into smaller metrics to be used for computational comparisons.

Once the data metrics are obtained, there are two major uses for it – verification/identification and comparison. The first consists of the verification or identification of an individual's identity. This is the verification that a person is who they claim to be (one-to one comparison) or their identity may be determined automatically from a list of known individuals (one-to-many comparison). The verification/identification process is primarily used to control access to secure facilities or networks. The other use is comparing individuals passing through a certain area such as an airport terminal to a 'watch-list' that may consist of a list of terrorists or wanted criminals (a many-to-many comparison). This application is far more complex because it requires collecting data from many subjects simultaneously and rapidly checking them against a database that could potentially be very large. Additionally, such a system is only useful if subjects are unaware that they are being observed, so it must be able to scan large areas that could potentially contain multiple subjects at the same time. By taking all of these individual elements into consideration, this research has developed a system that requires no user interaction and could be used in 'watch-list' application to monitor for suspects.

## 2. Project Description

Facial recognition is currently the most promising covert biometric system primarily because the face is the most distinct human trait that can be seen from a distance. There are many available schemes for facial identification currently being used, but generally speaking, the better the image quality (resolution, lighting, and orientation), the better the recognition rate. Currently several cities and airports have implemented security camera networks which seek to use stationary cameras to obtain images of individual's faces in order to screen them against a watchlist.


**Figure 1: Terrorists in London Subway on July 7th 2005**

These systems have had few successes, such as the capturing of the July 7, 2005 London Subway bombers on film, but for the most part have been ineffective at identifying individuals. For instance, the London Borough of Newham has had a facial recognition system built into their borough-wide Closed Circuit Television (CCTV)

system for several years, and has yet to identify a single suspect on the watch-list, even with several of the known suspects actually residing in the borough.

The primary problem with these CCTV systems is the quality of the images they take; security cameras are generally of low resolution, which greatly restricts the accuracy of a facial recognition algorithm. This project seeks to resolve this problem by developing an advanced security camera unit which actually contains two cameras; a low resolution web camera, and a high resolution still-frame digital camera. The block diagram in Figure 2 shows the system.

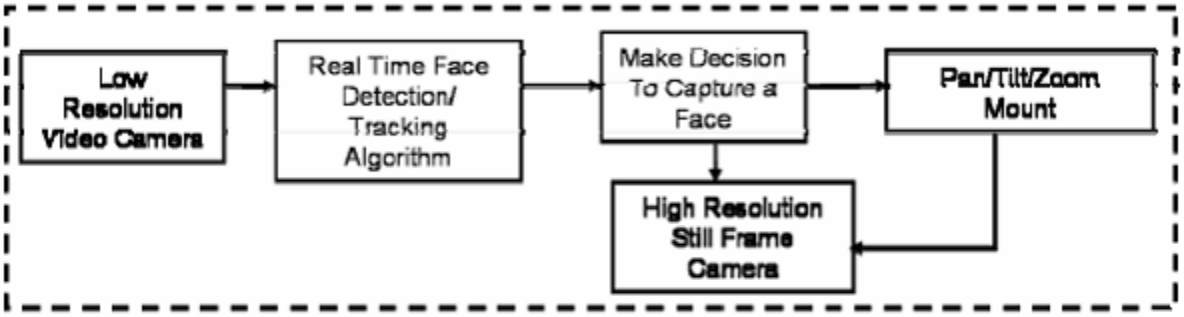


**Figure 2: Security Camera System**

The low resolution camera consists of a Logitech Quickcam Pro 5000 (Figure 3). This camera has proven ideal for this purpose because it combines good resolution (640 x 480 pixels), fast frame rates (30 frames per second), and a wide field of view (82 degrees) in a small robust package. The Quickcam is stationary, and surveys the entire area of interest at once, continuously monitoring for movement. When movement in the field of view is detected, the camera output is processed for face detection and localization.

**Figure 3: Quickcam Pro 5000**

The high resolution camera consists of a 6 mega-pixel (a mega-pixel is one million pixels) Canon Powershot S3IS (Figure 4). The camera is mounted on a high speed Directed Perception Pan/Tilt mount which can pan (left/right) a full 360 degrees and tilt (up/down) 180 degrees. The camera takes still-frame images which could be used for facial recognition processing. Producing these high resolution images is the primary purpose of the system.


**Figure 4: Canon Powershot S3IS**

With the detail provided in a 6 mega-pixel image, far more advanced facial recognition processing is possible than with low resolution video. The most promising of these higher resolution techniques is Skin Texture Analysis (STA) developed by the

Identix Corporation, which realizes much higher recognition rates than was possible with older techniques, such as Eigenfaces [3].

## 3. Interfacing of Components

The first major hurdle in developing a system that combines various off-the-shelf components into a single working unit is interfacing. Both the high resolution camera and the mount required separate controls to be developed in order to meet their desired functionality. This was accomplished in MATLAB.

Control of the Pan/Tilt mount (Figure 5) was accomplished using the computer's serial port. The serial port is used to send commands to the Pan/Tilt mount's controller, and allows control of both the movement of the mount and all of its settings, such as slew rate and acceleration.

**Figure 5: Directed Perception Pan/Tilt Mount**

Controlling the Canon Powershot proved to be much more difficult. The primary reason for this is that the Canon Software Development Kit (SDK), which provides a basic framework for communicating with their cameras was written in the C programming language. In order to use the camera, all of the control functions had to be ported into MATLAB using MEX files, which can interface C code with MATLAB.

MEX files are a specific type of MATLAB file which are written in C, but can compile and run in MATLAB. MEX files for initializing the camera, changing camera settings, and controlling the shutter were created. But the actual transfer of images from the high resolution camera to the computer over the USB (Universal Serial Bus) cable proved problematic, as the Canon framework for doing this could not be ported into MATLAB. As such, new functions were written in C which transferred the image data from the camera over the USB cable in individual packets. Packet sizes in initial testing were limited to only 1024 bytes, which resulted in very poor transfer speeds for images (upwards of 5 seconds). After several adjustments and revisions to the transfer protocol, 128 kilobyte packets were used successfully, which increased transfer speeds substantially (approximately one second for a one megabyte image). The complete assembled system can be seen in Figure 6.

**Figure 6: The System**

# 4. Foreground Extraction from Live Video

The first step in detecting motion in video is extracting regions of interest from the background. There are two approaches to this; motion segmentation and background subtraction. Motion segmentation is implemented by finding the absolute difference between the current frame and a previous frame and then thresholding the result. The equation for doing this is shown in equation (5-1) where pixel $x_k(m,n,c)$ in frame $k$ is subtracted from a frame $l$ frames ago, and thresholded using a threshold value $T$. Here, $m$ and $n$ represent the row and column numbers, and $c$ represents the three color planes of red, green, and blue.

$$\sum_{c=1}^{3} \left| x_k(m,n,c) - x_{k-l}(m,n,c) \right| > T \qquad \text{(5-1)}$$

An example of motion detection from the system is shown in Figure 7. As can be seen, there is a substantial border around the people; this is due to the morphological techniques used on the areas where motion was detected. The morphology consists of erosion to eliminate noise, and then dilation to expand the areas of motion. This is done because it was determined in testing that it is better to get extra area surrounding detected motion in this step, rather than risk losing areas that contained people's faces due to processing tolerances. Additionally, it can be seen that there are some areas where motion is detected that are in fact not moving. This is, for the most part, due to variations in lighting and video noise. While motion segmentation is computationally efficient and very adaptable to changing backgrounds, testing has shown that it has limitations. Variations in lighting, video noise, and stationary foreground objects can cause it significant problems.

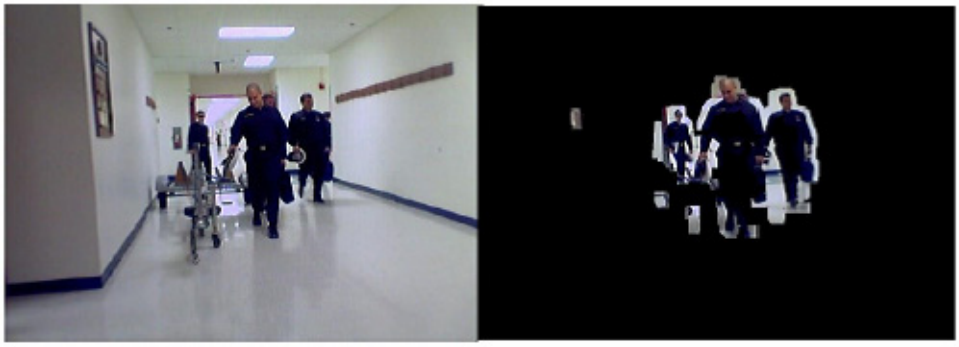

**Figure 7: Motion Segmentation**

Background subtraction, while more computationally intensive, can be more reliable. This technique requires building a composite background, which can be done in many different ways, each which have their own advantages and disadvantages. The most common methods are Kalman filtering [4], mixture of Gaussians [5], and median filtering [6]. For the purposes of this research, none of these techniques were computationally efficient enough to be applied, so a new technique is being developed that builds on these techniques. The base premise of the approach is that pixel values for background will reoccur quite often over time, with slight variations due to lighting. As such, a composite background is built where weights are assigned for the values of each pixel, depending on frequency of occurrence over time. These weights are adjusted in each frame according to equation (5-2), where $w_i$ specifies the weight for intensity value $i$, $L$ is the learning rate, and $K$ is 1 for the new value and 0 for others.

$$w_i = w_i + \frac{1}{L}(K - w_i) \tag{5-2}$$

As new values appear, they are added to the list of weights, and so over time, the most commonly occurring values for each pixel will be given large weights. If a foreground

object moves into the scene, its weight value will be extremely low, and thus can be segmented. This technique is a work in progress, and has yet to be fully implemented in the system.

## 5. Color Segmentation for Skin Detection

Once foreground regions are segmented, these regions are put through further processing to determine where skin is located in the scene. Skin detection is based on the principle that in certain color planes, skin values occupy a very narrow range of values, even when different races are considered. After extensive testing, the LUV color space was chosen because it allowed for the most accurate segmentation, while still using integer values (which allows for a fast processing technique that will be shown later). A pixel in the LUV color plane contains three values; one for luminance (brightness), and two for chrominance (color). Experimental testing showed that a pixel at location $(x,y)$ could be reliably called skin tone if it satisfied equation (6-1).

$$(140 < L_{(x,y)} < 230) \cap (100 < U_{(x,y)} < 118) \cap (140 < V_{(x,y)} < 158) \qquad \textbf{(6-1)}$$

In order to implement the skin segmentation in a manner as efficient as possible, the number of comparisons to determine if a pixel is skin must be kept as low as possible. This is necessary because every video frame contains 640x480 pixels, or 307,200 pixels. If equation (5-3) were implemented in a brute force approach, each pixel would require 8 comparisons, or 2,457,600 comparisons per frame. This proved to be too slow, since 30 frames per second were being processed, so a new faster approach was implemented. In this approach, the range of values that defined as skin are put in three data structures, one for each portion of the color space. A graphical representation of these structures is seen in Figure 8. The structures contain a logical 1 or 0 (corresponding to whether or not it

falls in the range for skin) for every possible L, U, and V value (integers with value 0 to 255). These structures allow for much faster skin detections, as it only requires three quick "AND" operations to determine if a pixel lies within the skin region. This reduces the maximum number of computations per frame from 2,457,600 to 921,600. In trials, this amounted to a reduction of about .05 seconds per frame (a significant amount of processing time saved for real time video).

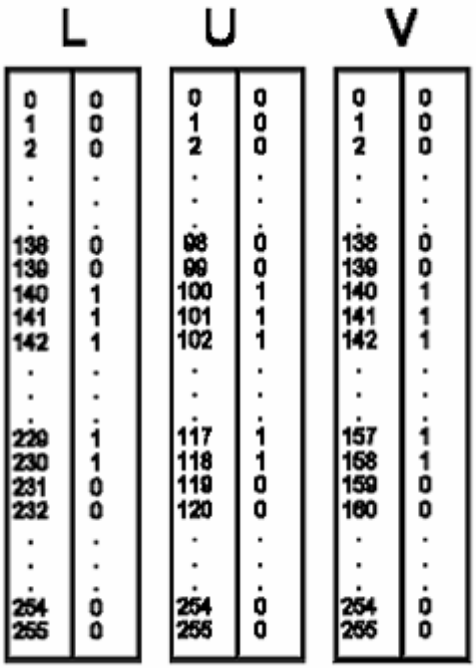


**Figure 8: Skin Segmentation Data Structure**

Figure 9 shows the motion detected image from the previous section on the left, and on the right a binary image which shows in white where skin has been detected. As can be seen, there are some errors in the system, as the colors of certain background areas fell within the range of values specified as skin tones. This is not a significant problem, and will be solved once the new background segmentation code is implemented. Aside from that one error, Figure 9 shows that at this point the video frame has been reduced to

only two types of objects; hands and faces. This is where the next step in the algorithm, the neural network classifier, comes into play.

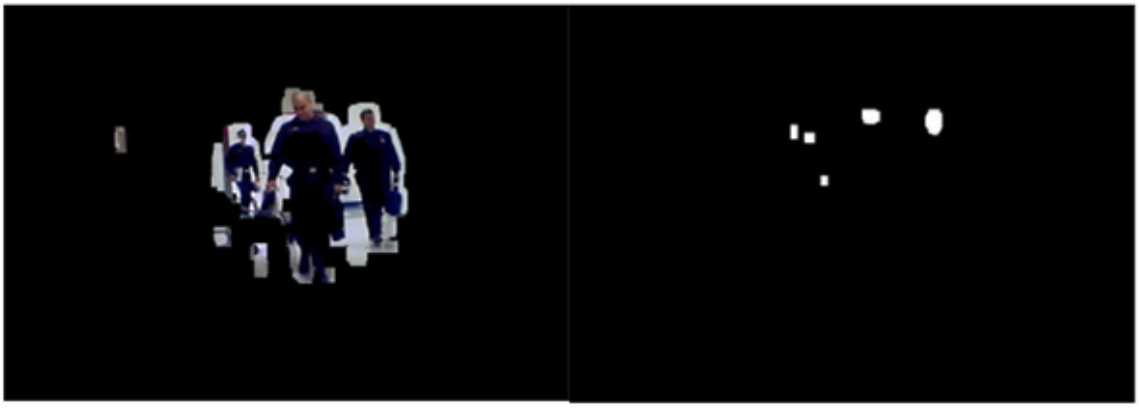

**Figure 9: Skin Segmentation**

## 6. Neural Network for Window Classification

At this point the video frame has been reduced to a few windows which contain either faces or other objects (such as hands). Once these windows are extracted, they must be classified as either faces or non-faces. To do this a multilayer perceptron neural network classifier is employed. Neural networks, named after their resemblance in functionality to the human nervous system, consist of large numbers of simple processing elements called neurons, which exhibit complex global behavior based on their connections and individual neuron parameters. Neural networks are used primarily to solve problems for which no mathematically optimal solution exists; such as determining whether or not an image contains a face.

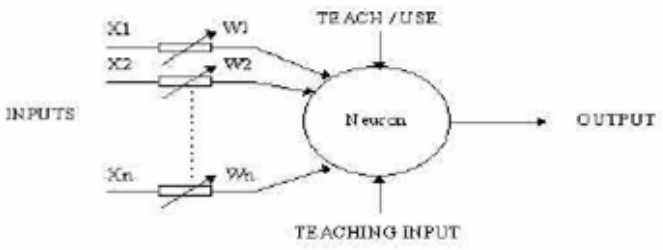


**Figure 10: A Neuron**

Each individual neuron in a neural network is simple in nature; it consists of a summation and a thresholding function. An example of a neuron is shown in Figure 10. Each input $X_i$ is multiplied by a weight $W_i$ and these products are summed in the neuron. This value is then thresholded by a sigmoid function, and the output, either a one or zero, is sent to its output, which may be the next layer of nodes. The type of neural network employed in this research is called a multilayer perceptron feed-forward network. This type of neural network is characterized by two principal elements; all of the inputs are connected to every hidden node, and information only goes forward (i.e. there is no feedback). This type of network was chosen because of its speed; since there is no feedback, all computations can occur simultaneously, allowing very fast classification. An example of a feed forward network similar to the one used is located in Figure 11.

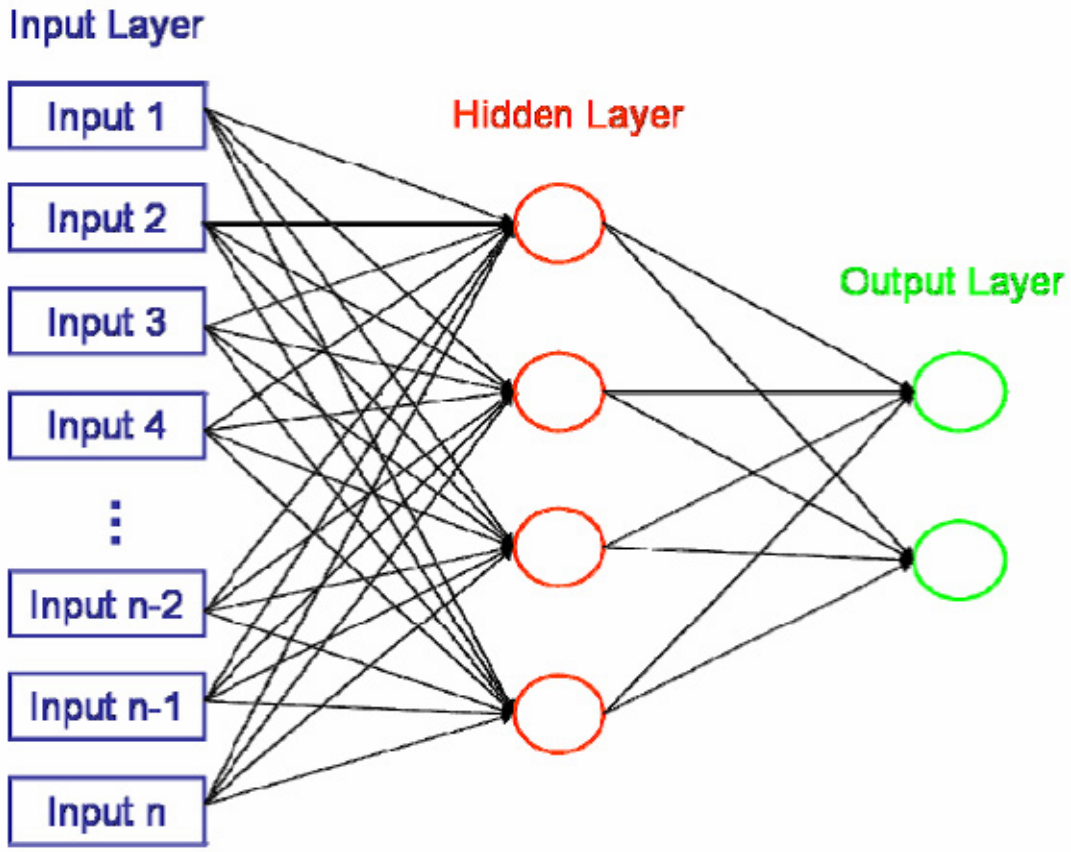


**Figure 11: A Feed Forward Neural Network**

Figure 11 shows a neural network that has n input values presented at a time to each of four neurons in a "hidden" layer. The output of these hidden neurons is passed to each of two neurons in the output layer.

As discussed earlier, each neuron or "node" has weights which determine the output for any given set of inputs. In order to determine the values of the weights, and thus get the correct output, the network must be trained. This is accomplished by building a set of training data, which contains thousands of examples of faces and non-faces. This training data set was built by recording video of people walking down a hallway, and then using the previous two steps, motion segmentation and flesh detection to extract

windows. These windows were then all manually classified by the author as faces or non-faces. In order for the windows to be processed by the network code, they must be changed from image matrices to one-dimensional vectors as illustrated in Figure 12.

Once each window is classified and transformed into a vector, a matrix is created in which each row is an represents a window's vector, with the first column containing the correct classification (a 1 for face, or a 2 for non-face). This matrix is then used iteratively to find the best weights for the hidden nodes so that the neural network obtains the best classification performance possible (since the correct classification is known during training). One of the key variables in training is the number of hidden nodes; various numbers of nodes were tested in order to determine the number of hidden nodes that gave the best accuracy. Six hidden nodes worked the best, giving fast computation times and over 96% accuracy on the training set.

One of the other key variables is the number of input values passed into the neural network for each classification (that is, number of elements in each input vector). For this research, this amounted to the number of pixels used for a face window. Various window sizes were tested, ranging from as small as 6x6 up to 30x30. It was found that increasing resolution past 12x8 actually resulted in a decrease in performance (both in efficiency and accuracy). While at first this may seem counter intuitive, it is a common occurrence in neural networks. It is primarily due to the fact that faces only have so many distinguishing features common to all of them; a round shape, light areas on the nose and forehead, and dark areas around the eyes and mouth. Once the resolution is increased beyond a certain point, the extra information provided by the extra pixels is spurious at best, and often detrimental to performance. This entire process is illustrated in Figure 12.

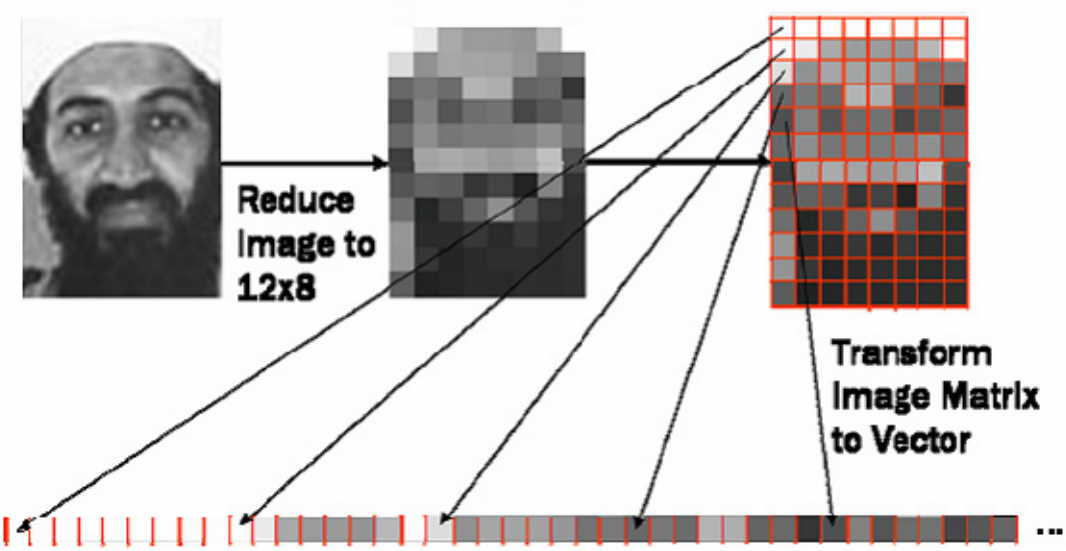


**Figure 12: Resizing and Transformation**

## 7. Overall Operation

Once a window is classified as a face, pertinent information (centroid location, size, and speed) about the window is stored in a data structure. Only location and size are stored in the initial detection, with speed being stored after the same face is detected in four successive frames. Speed is determined by a weighted average of differences between the centroids from successive frames. The equation for doing this is shown in equation (8-1), where $n$=1 is the most recent frame, and $k_n$ are the weight average constants.

$$\Delta x = \sum_{n=1}^{3} k_n (|x_n - x_{n+1}|)$$
$$\Delta y = \sum_{n=1}^{3} k_n (|y_n - y_{n+1}|) \qquad (8\text{-}1)$$
$$k_n = [.5, .3, .2]$$

The weighted average allows for changes in speed, while preventing excessive spikes due to errors in window centering. Once a speed is determined, the face is queued for capture. When the face reaches the beginning of the queue, the pan/tilt mount is moved to its next location (as predicted by the face's speed), and a high resolution image is taken. This method is particularly robust to error because a window is only queued for imaging if it occurs four times in a row. For something other then a face to be queued, it would have to be falsely detected in four successive frames, which testing has shown is extremely unlikely. A flow chart of the entire system operating is shown in Figure 13. The entire process is run from one MATLAB graphical user interface. The current version of the code encompasses over 50 files, containing approximately 5000 lines of code.

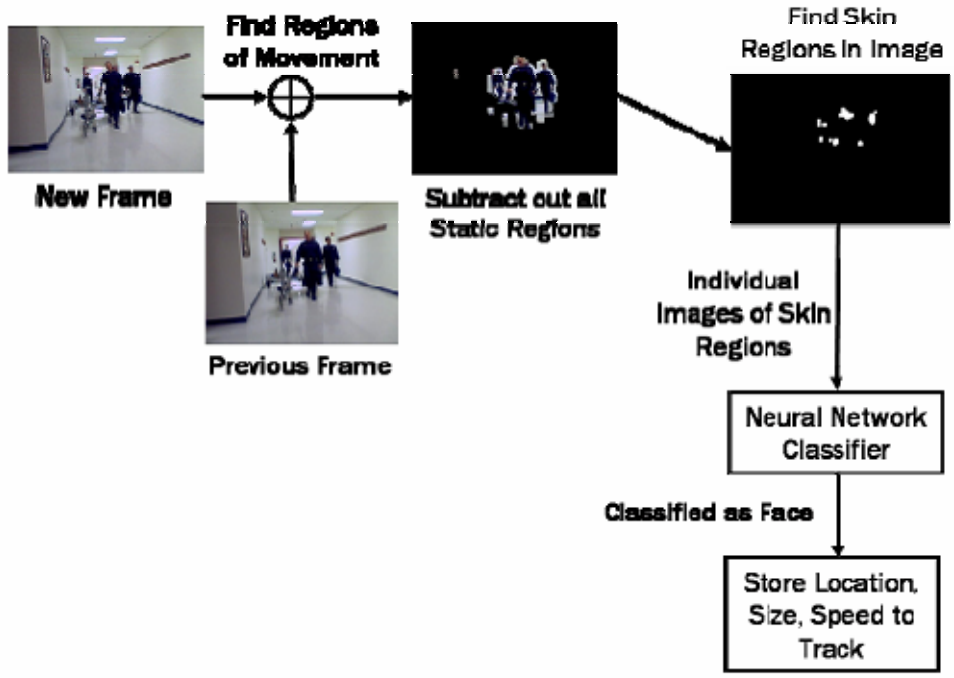


**Figure 13: Overall Outline of System Operation**

## 8. System Testing

Due to recent events involving human subject testing issues, testing on controlled human data has ceased. This will be addressed in the Spring, with a shift to other objects as necessary (due to human subject testing concerns). Regardless of what the system is being tested on, final results will include measures of response times, capture rates, and system errors.

### Response Times

The system has two different response times. The first is a measure of how long it takes the system to find an object of interest once it enters the scene. This is important primarily in scenarios where objects are present in the scene for only a short amount of time. The other measure of response time is how long it takes to capture a high resolution image of an object, once it has been detected. This is one of the most important variables

to measure, since it determines how many objects can be in the scene before they begin to get missed due to the system being too slow.

### Capture Rates

This variable measures the time between successive captures with the high resolution camera. This is measured by finding the time between captures when the system is presented with many detections.

### System Errors

Errors come in two main forms; capture errors and classification errors. Capture errors occur when the system correctly detects an object, but then fails to take the high resolution image of the object. This type of error would be primarily due to bad predictions of future locations.

Classification errors come in two forms; false positives and false negatives. False positives occur when the system classifies something as being an object of an interest when in fact it is not. False negatives occur when the system fails to correctly classify an object of interest. False positives and false negatives are the most reliable indicators of the performance of the neural network classifier.

## 9. Conclusions

Biometric identification, public surveillance, and unmanned vehicle systems all currently suffer from one common handicap; their use of low resolution video. This severely restricts their ability to accurately achieve their respective goals. The conclusion of this research will be a functioning system that addresses this problem by allowing the accurate and efficient capture of high resolution images of objects. The ability to detect

any type of predefined object and autonomous nature of the end-system will allow for simple application to all three of these fields, and more.

## 10. Spring Semester Deliverables

Due to the human subject testing issues, the spring deliverables may change, but currently the goal is the complete system that is able to be rapidly adjusted to varying applications. In order for this to be accomplished, there are three main areas that must be addressed. The first is the development of the background segmentation algorithm discussed in Section 4. While the current method of motion detection works well enough in controlled environments, a more robust algorithm is necessary for the system to be useful in uncontrolled real-world environments. The second deliverable is the development of an efficient way to adjust the system's classification algorithm to different applications. Currently this process is extremely time intensive, and requires large amounts of human interaction. The final version of the system will allow for automization of most of this process. The final deliverable consists of all the testing discussed in Section 8 and corresponding analysis. This will allow us to evaluate the system's overall performance and viability in different real-world applications.

# 11. Works Cited

[1] "Human Research Protection Program", *SECNAV Instruction 3900.39D*, 2006 Nov 3, Available HTTP: http://www.onr.navy.mil/sci_tech/34/docs/secnavinst_3900_39d.pdf

[2] P. Hallinan, G.G. Gordon, A.L. Yuille, P. Giblin, and D. Mumford, *Two and Three Dimensional Patterns of the Face*, Natick: A K Peters, 1999.

[3] W.Y. Zhao, "Image-Based Face Recognition: Methods and Issues", [Online Document], cited 2006 Jan 14, Available HTTP: http://www.face-rec.org/interesting-papers/General/Chapter_figure.pdf

[4] C. Wren, A. Azabayejani, T. Darrel, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, pp. 780-785, July 1997.

[5] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence* (UAI-97), pp. 175-181, Morgan Kaufmann Publishers, Inc., (San Francisco, CA), 1997.

[6] Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from videos," *Proceedings of IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.

## 12. Works Consulted

1. B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge: Cambridge University Press, 1996.

2. T. Masters, Signal and Image Processing with Neural Networks, New York: Wiley, 1994.

3. L. Li, W. Huang, I. Yu-Hua Gu, and Q. Tian, " Statistical Modeling of Complex Backgrounds for Foreground Object Detection," IEEE Transactions On Image Processing, vol. 13, no. 11, Nov., pp. 1459-1472.

4. R. Duda, P. Hart and D. Stork. Pattern Classification 2nd Ed., New York: John Wiley and Sons, 2001.